IIASA PROCEEDINGS SERIES

Decision Support Systems: Issues and Challenges

Göran Fick and Ralph H. Sprague, Jr., Editors

International Institute for Applied Systems Analysis

IIASA PROCEEDINGS SERIES

VOLUME 1 — CARBON DIOXIDE, CLIMATE AND SOCIETY
Proceedings of an IIASA Workshop Cosponsored by WMO, UNEP, and SCOPE, February 21–24, 1978
Jill Williams, Editor

VOLUME 2 — SARUM AND MRI: DESCRIPTION AND COMPARISON OF A WORLD MODEL AND A NATIONAL MODEL
Proceedings of the Fourth IIASA Symposium on Global Modelling, September 20–23, 1976
Gerhart Bruckmann, Editor

VOLUME 3 — NONSMOOTH OPTIMIZATION
Proceedings of an IIASA Workshop, March 28–April 8, 1977
Claude Lemarechal and Robert Mifflin, Editors

VOLUME 4 — PEST MANAGEMENT
Proceedings of an International Conference, October 25–29, 1976
G.A. Norton and C.S. Holling, Editors

VOLUME 5 — METHODS AND MODELS FOR ASSESSING ENERGY RESOURCES
First IIASA Conference on Energy Resources, May 20–21, 1975
Michel Grenon, Editor

VOLUME 6 — FUTURE COAL SUPPLY FOR THE WORLD ENERGY BALANCE
Third IIASA Conference on Energy Resources, November 28–December 2, 1977
Michel Grenon, Editor

VOLUME 7 — THE SHINKANSEN HIGH-SPEED RAIL NETWORK OF JAPAN
Proceedings of an IIASA Conference, June 27–30, 1977
A. Straszak and R. Tuch, Editors

VOLUME 8 — REAL-TIME FORECASTING/CONTROL OF WATER RESOURCE SYSTEMS
Selected Papers from an IIASA Workshop, October 18–20, 1976
Eric F. Wood, Editor, with the Assistance of András Szöllösi-Nagy

VOLUME 9 — INPUT–OUTPUT APPROACHES IN GLOBAL MODELING
Proceedings of the Fifth IIASA Symposium on Global Modeling, September 26–29, 1977
Gerhart Bruckmann, Editor

VOLUME 10 — CLIMATIC CONSTRAINTS AND HUMAN ACTIVITIES
Task Force on the Nature of Climate and Society Research, February 4–6, 1980
Jesse Ausubel and Asit K. Biswas, Editors

VOLUME 11 — DECISION SUPPORT SYSTEMS: ISSUES AND CHALLENGES
Proceedings of an International Task Force Meeting, June 23–25, 1980
Göran Fick and Ralph H. Sprague, Jr., Editors

IIASA PROCEEDINGS SERIES

Volume 11

Decision Support Systems: Issues and Challenges

DECISION SUPPORT SYSTEMS: ISSUES AND CHALLENGES

Proceedings of an International Task Force Meeting
June 23–25, 1980

GÖRAN FICK
International Institute for Applied Systems Analysis
Laxenburg, Austria

RALPH H. SPRAGUE, JR.
University of Hawaii
Honolulu, Hawaii

Editors

PERGAMON PRESS

OXFORD · NEW YORK · TORONTO · SYDNEY · PARIS · FRANKFURT

U.K.	Pergamon Press Ltd., Headington Hill Hall, Oxford OX3 0BW, England
U.S.A.	Pergamon Press Inc., Maxwell House, Fairview Park, Elmsford, New York 10523, U.S.A.
CANADA	Pergamon of Canada, Suite 104, 150 Consumers Road, Willowdale, Ontario M2J 1P9, Canada
AUSTRALIA	Pergamon Press (Aust.) Pty. Ltd., P.O. Box 544, Potts Point, N.S.W. 2011, Australia
FRANCE	Pergamon Press SARL, 24 rue des Ecoles, 75240 Paris, Cedex 05, France
FEDERAL REPUBLIC OF GERMANY	Pergamon Press GmbH, 6242 Kronberg-Taunus, Hammerweg 6, Federal Republic of Germany

First edition 1980

British Library Cataloguing in Publication Data

Decision support systems. - (International Institute for Applied Systems Analysis. IIASA proceedings series; vol. 11).
1. Management - Data processing
2. Decision-making - Data processing
I. Fick, Göran II. Sprague, Ralph H
III. Series
658.4'03'02854044 HD30.23 80-41670

ISBN 0 08 027321 1

Printed and bound in Great Britain by William Clowes (Beccles) Limited, Beccles and London

FOREWORD

The IIASA Management and Technology Area is studying, among other topics, the interplay between management, managers, and technological change. One major study area has been the nature of the innovation process and the strategies open to policy makers who want to improve its social utility. Another focus of interest has been the management of industrial complexes with low risk but potentially large hazards. In the beginning of 1979 the rapidly changing field of information technology was selected as a third research area. Underlying the research is the realization that rapid technological development forces management to make decisions in a highly turbulent environment.

The Management and Technology Area has selected some small but essential areas in this vast field for closer examination. The concept of "decision support systems" (DSS) is one in which an intensive and potentially important development is taking place. The MMT task on "the interface between managers and their computer tools," headed by Göran Fick, organized an invited meeting for June 23-25, 1980 to identify the state of the art in DSS and reach a better understanding of future research needs. The material presented in these Proceedings makes the findings of the meeting available for a larger audience.

Alec Lee
Chairman
Management and
Technology Area

CONTENTS

INTRODUCTION

Continued pressure to improve the efficiency and effectiveness of organizations has led managers to seek help from information technology. The assistance provided by this technology has evolved from mechanized paperwork in the form of early data processing systems, through integrated files and inquiry systems of the MIS era, to the current emphasis on office automation. Growing evidence suggests that a class of problems and needs in organizations has not been susceptible to traditional information support in the past; this problem area involves decision making at all levels of organizations by individuals and groups.

Recent advances in information technology seem to be promising as aids for decision making; these include micro-computers, color graphics, and telecommunication networks. However, early attempts to "throw technology at the problem" have revealed that such a direct approach has serious limitations. In fact, it appears that a significant integration of advances in both technology and a set of related disciplines will be required, in order to apply information technology intelligently and effectively to the class of unmet problems and needs facing decision makers in organizations. This is the goal of developers of Decision Support Systems (DSS).

This book reports on a three-day meeting on Decision Support Systems held at the International Institute for Applied Systems Analysis (IIASA). IIASA's interest in sponsoring the meeting was spurred by several factors. First, the term DSS clearly is used in a wide range of contexts; we hoped to develop a deeper understanding of the term and the new field to which it refers. Second, we felt that ongoing work in the DSS field would be enhanced by interaction between professionals who had been working on such systems and people from fields that function as "resource disciplines" for DSS. Finally, we wished to bring professionals from several nations together, from the east as well as the west, to share experiences and to assess the viability of the DSS concept in different cultures.

Before summarizing the format and general results of the meeting, we would like to communicate two significant and

unanticipated conclusions that resulted from the discussions. The first is the concern, expressed by many of the participants, that the application of technology to human needs and problems be conducted with humility within the moral and ethical context of scientists and professionals. Moral responsibility of professionals is a prerequisite for the positive outcome of any activity in the management information systems field. This is especially true in the case of DSS, because its applications are decision problems that are not well understood by the group of people involved; such problems cannot be "solved" by a single systems analysis effort or a highly structured computerized decision aid. Openness and willingness to take part in a learning process and to redefine system boundaries, structures, and methods are essential for DSS professionals.

A basic moral issue is whether or not to recommend *computerized* support systems. It is necessary to judge whether computer support is appropriate, and, more fundamentally, whether the organization will benefit in a wide sense from the introduction of such a system. For example, it could happen that a given computer system is appropriate for a certain task, but it may focus attention on this task to such a degree that other tasks, not included in the computer system, are neglected.

The second unanticipated result was the realization that the systems analysis and development approach underlying the class of systems called DSS may be significantly different from traditional systems analysis paradigms. A DSS requires a development strategy that differs from the traditional analysis-design-programming-implementation regime. The systems requirements for a DSS are often difficult or impossible to define. Managers facing a complex situation do not generally know what they want or need. The absence of a comprehensive theory of organizational behavior (decision making) means that no one else knows either.

The prescription offered by DSS is a shorter feedback loop between the designer-developer and the user, with frequent repetition of a simple development cycle. A DSS is built so that it can be easily changed with each cycle. The approach calls for the initial identification of a small, critical subproblem or set of decisions. A small but complete system is then developed to support the set of decisions initially perceived by designer and user. This system is used and evaluated for a relatively short period of time. The evaluation guides the next cycle of analysis, changes, additions, and deletions that expand or redirect the system's capabilities.

The designer must be humble enough to recognize the limitations of his/her own favored system structures, and also must be prepared to advise the use of other structures, systems, and technologies, or to terminate the ongoing development activity at each cycle. The application of mechanistic "solutions" to poorly understood problems of managerial decision making must be prevented in each cycle.

This development strategy is different in some subtle but significant ways from other approaches that sound similar. The

initial system is not a prototype or pilot test; it is a real and useful operating system. The approach requires more than just management participation in the design. The entire process is driven by the manager/user who *is* the builder and designer. The system analyst is the catalyst and facilitator, who helps to implement the user's specifications.

The traditional one-shot systems analysis strategy is thus challenged by DSS. The DSS approach recognizes that there is a class of underspecified problems or issues for which cycles of repetitive systems analysis and learning generated by system use is a better alternative.

SUMMARY OF THE PAPERS

Our strategy for the meeting was to permit professionals from several disciplines and from several countries to interact and to exchange their views on and expertise in DSS. Some participants had been studying DSS from a research viewpoint, and others had experience developing DSS. As well, key people from several resource disciplines, who were only vaguely aware of the DSS area, attended the meeting.

During the first half of the meeting, representatives from each of these groups presented invited position papers. The Sprague and Keen papers were designed to establish a context and frame of reference for the remaining papers and the discussions. Sprague presents a conceptual model of DSS derived from hardware and software developments and observations of operating DSS. Keen traces the historical development of the field and presents a view of organizational and behavioral roles in the development of DSS. This paper has an extensive bibliography and provides a list of about 30 specific DSS cases in an appendix. A second appendix cross-references these cases with a set of characteristics that tend to distinguish DSS from other computer-based systems.

The second set of invited papers comes from several related disciplines that promise to make significant contributions to the development of DSS. Here the authors interpret some of the theory and development of their own fields in the light of DSS work, and give some insight into potential, but as yet underdeveloped, contributions to DSS. For example, Huber's paper on organizational sciences reveals that DSS must accommodate *organizational* decision making as well as *individual* decision making. Frank Manola's paper summarizes much of the current development in the database field, and reveals the lack of data models for handling time series and judgmental data. Sagalowicz touches on the potential significance of Artificial Intelligence (AI) in dealing with knowledge representation for DSS. The paper by Flores raises basic questions about the nature of a "decision." Flores argues that discussions of decision making have been too preoccupied with the choice between alternatives. In Flores' view, decision making is driven by five basic types of "speech acts": directives, commissives, assertions, declarations, and

expressions. Unless an information system accommodates all these "speech acts," it may be susceptible to failure.

Observations on several existing systems that have a DSS orientation are presented in the final set of papers. These application experience papers demonstrate the viability of the DSS concept in many cultures and reveal a remarkable similarity in approaches in diverse countries.

THE GROUP SESSIONS

To create an environment for interaction, four discussion groups were formed, and the last half of the meeting was devoted to group discussion and reports. The objective was to bring the collective intelligence of each group to bear on the "issues of DSS." In order to provide variety and breadth, the four groups were given the following broad charter:

> The participants are asked to discuss the issues and problems facing DSS in order to better define its aims, to identify the means and barriers to its successful implementation, and to outline programs for its continuing development.

It was hoped that this charter would trigger a wide range of responses from the groups; it is clear from the group reports that this intent was fully realized! The reports range from a pragmatic, directed set of steps for the development of DSS, to a wide-ranging philosophical discussion of the ethics of technology application. The final remarks by M.A.H. Dempster provide an excellent synthesis of the reports of the four groups and summarize the overall outcome of the meeting.

SUMMARY

The broad objectives set for this meeting were realized in a variety of ways. Virtually all the participants testified that they had gained a deeper understanding of DSS, the role it can play in assisting managers in organizations, and the need for further development in key areas. We are pleased to share the product of their work and trust that readers will find it stimulating and useful.

Göran Fick
Management and Technology Area
International Institute for Applied Systems Analysis

Ralph H. Sprague, Jr.
Department of Decision Sciences
University of Hawaii

A FRAMEWORK FOR RESEARCH ON DECISION SUPPORT SYSTEMS

Ralph H. Sprague, Jr.
University of Hawaii

INTRODUCTION

The purpose of the first two papers written by members of our Task Force on Decision Support Systems is to establish a common frame of reference for the following papers and discussion. We shall not attempt a comprehensive definition of the term 'decision support systems' (DSS) at this time, although many may feel the need for it; because of the complexity of the subject, such attempts are frustrating and doomed to failure. Rather we will consider several alternative views of DSS, charting characteristics and attributes, as a way of developing a better understanding of the subject.

Let us begin with as broad a view as possible, to establish the relationship of DSS to other areas of activity. DSS clearly lies within the area of *information systems in organizations*, whose broad charter can be expressed in the following way:

> Dedicated to improving the performance of knowledge workers in organizations through the application of information technology.

A closer look at the terms of this charter may clarify the role of information systems in organizations:

(1) Improving performance is the ultimate objective of information processing, not the storage of data, the production of reports, or even "getting the right information to the right person at the right time." The ultimate objective must be viewed in terms of the ability of information systems to support the improved performance of people in organizations.

(2) *Knowledge workers* are the clientele of information systems professionals. Knowledge workers include managers,

professionals, staff analysts, and clerical workers whose primary job responsibility is the handling of information in some form.

(3) *Organizations* are the context of information systems work. We are not primarily concerned with the information handling requirements for launching a satellite, plotting the trajectory of a missile, or calculating the time of the next solar eclipse. The focus is instead on information handling in goal-seeking organizations of all kinds.

(4) *The application of information technology* is the challenge and opportunity facing the information systems professional to meet the objective of improving performance in organizations.

Two caveats are appropriate at this point. First, the word "technology" should be interpreted broadly to include the full range of technical tools—including computers, communication capability, management science models, etc. Second, as responsible informations systems professionals, we must not be "technology pushers," who diligently seek problems which are susceptible to the tools they know how to use. In fact, there is ample evidence that managers and organizations are in desperate need of creative approaches to applying the technology which is developing at an increasingly rapid rate.

Our understanding of how to apply information technology has, of course, evolved as the technology has grown. The current interest in DSS is seen by some as a repeat of the enthusiastic reception given MIS concepts about a decade ago. That enthusiasm was followed by inevitable disenchantment, as MIS failed to perform as expected. Many of us fear a recurrence of this cycle with the current use of DSS as a "buzz word." The only hope for avoiding this wide swing in expectations is a realistic appraisal of what the DSS concept is, and what it can do. To begin, let us consider a popular way of visualizing the focus of information options in organizations.

A triangle was used by Robert Head in the late 1960s as a visual model of MIS. It has become a classic way to view the dimensions of an information system. The vertical dimension represented the levels of management and the horizontal dimension represented the main functional areas of the business organization. Other authors later added transactional processing, using it as a base on which the entire system rested. The result was a two-dimensional model of an MIS in the broad sense—i.e., the total activities which comprise the information system in an organization. Figure 1 shows a further extension of the basic triangle to help conceptualize the potential role of DSS. The depth dimension indicates the major technology "subsystems" which provide support for managerial activities: the structured and required reporting system (including what is often called MIS), data communication and clerical systems (including word processing and office automation), and DSS.

To summarize this introductory section, DSS draw on transactions processing systems, and interact with the other technology

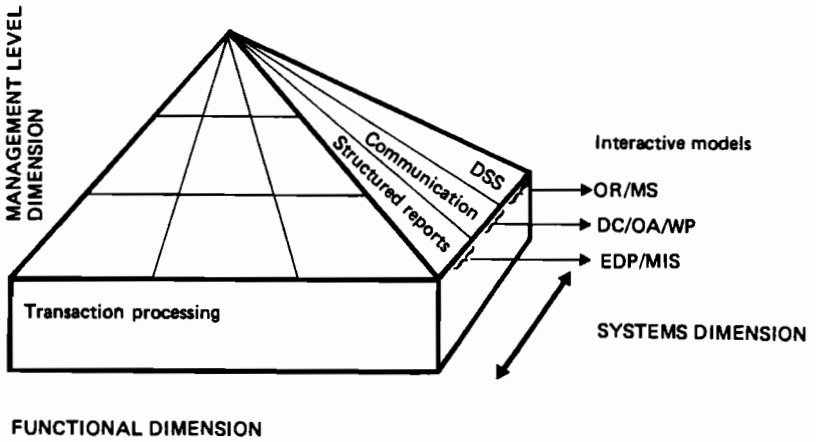


FIGURE 1 A visual model of the dimensions of an information system.

to support the decision-making activities of managers. DSS is not merely an evolutionary advancement of EDP and MIS, and it will certainly not replace either. Nor is it merely a type of information system aimed exclusively at top management, in cases in which other information systems seem to have failed. It is, rather, another powerful weapon in the information technology arsenal, aimed at improving the effectiveness of managers in organizations.

As we shall see, there are some subtle but significant differences between DSS and traditional EDP or so-called MIS approaches. Moreover, a new coalescence of all these information systems technologies is needed to satisfy a set of heretofore unmet needs. We are not yet sure how these technologies fit together, or which important problems need to be solved. Indeed, a large part of the purpose of this task force meeting is to address these questions.

To establish the context for our discussions, this paper first looks briefly at a definition of DSS and then considers three views which seem to characterize much of what is meant by the term in current usage. These three views form a framework for examining the key characteristics of DSS. The paper closes with a suggested list of research issues and methodologies which are needed for the further development of the subject area.

DEFINITIONS

The concepts involved in DSS were first articulated in the early 1970s by Michael S. Scott Morton; he used the term 'management decision systems' (Morton 1971). Later the term 'decision support systems' became current, as a few firms and a few scholars began to develop this area of research. DSS became characterized

as "*interactive* computer-based systems which *help* decision makers utilize *data* and *models* to solve *unstructured* problems." The unique contribution of DSS was felt to result from the key words given in italics. However, this definition proved so restrictive that few existing systems completely satisfied it. The province of DSS has been recently broadened by some authors to include any system that makes some contribution to decision making; this implies that the term can be applied to all but transaction processing. A serious definitional problem is that the words have a certain "intuitive validity." Any system that supports a decision (in any way) is a Decision Support System—obviously!

Unfortunately, neither the restrictive nor the broad definition help much, because they do not provide guidance for understanding the value, the technical requirements, or the approach needed for developing DSS. A complicating factor is that people from different backgrounds and contexts view a DSS quite differently. A manager and a computer scientist seldom see things in the same light.

THREE VIEWS OF DSS

There seem to be three levels from which a DSS can be viewed. These three levels are characterized by the type of computer hardware and software which is referenced by the DSS term, and the hierarchical specificity of task to which the DSS is applied. The system which actually accomplishes the work might be called the *specific DSS*. Such a system is a DSS "application," but with characteristics that make it significantly different from a typical data processing application. Here the DSS is the "software" that allows a specific decision maker or group of them to deal with a specific set of related problems. An early example is the portfolio management system (Gerrity 1971), also described in the first major book on DSS (Keen and Morton 1978). Another example is the police beat allocation system used on an experimental basis by the County of Santa Clara (Carlson and Sutton 1974). This allocation system allowed a master sergeant in the police force to display a map outline and call up data by geographical zone, showing police calls for service, activity levels, service time, etc. The interactive graphic capability of the system enabled him to manipulate the maps, zones, and data to try a variety of police beat alternatives quickly and easily. In effect, the system gave him tools to amplify his managerial judgment. (Incidentally, a later experiment attempted to apply a traditional linear programming model to the problem; the solution was less satisfactory than the one designed by the sergeant.)

The second level on which to consider DSS might be called the *DSS generator*. This is a package of related software which provides a set of capabilities to quickly and easily build a specific DSS. For example, the police beat system described above was built from the Geodata Analysis and Display System (GADS), an experimental system developed at the IBM Research Laboratory in San Jose (Carlson *et al.* 1974). By loading different maps, data, menu choices, and procedures (command strings), GADS was later used to build a *specific DSS* to support the routing of IBM copier

repairmen (Sutton 1978). The development of this new application required less than one month.

Another example of a DSS generator is the Executive Information System (EIS) marketed by Boeing Computer Services. EIS is an integrated set of capabilities which includes report preparation, inquiry capability, a modeling language, graphic display commands, and a set of financial and statistical analysis subroutines. These capabilities have all been available individually for some time; the unique contribution of EIS is in making them available through a common interface working on a common set of data. The extent to which the system incorporates the other attributes of a DSS is subject to the opinion of its users, but the structure and intent of EIS indicates that it could become a DSS generator, especially for financial decision making.

Most of the evolutionary growth leading toward DSS generators has come from special purpose languages. In fact, most of the software systems claiming to be DSS generators have evolved from enhanced planning languages or modeling languages, perhaps with report preparation and graphic display capabilities added. The Interactive Financial Planning System (IFPS) marketed by Execucom Systems of Austin, Texas, and EXPRESS, available from TYMSHARE, are good examples of this evolution. Here it is also appropriate to note the increasing interest in Application Development Systems (Zolliker 1980). In brief, we may describe a DSS generator as an *application development system* for that class of application which we are calling a specific DSS.

The third and most fundamental level of technology applied to the development of DSS may be called *DSS tools*. These are hardware or software elements which facilitate the development of a specific DSS or a DSS generator. This category of technology has seen the greatest amount of recent development, including new special purpose languages, improvements in operating systems to support conversational approaches, color graphic hardware and supporting software, etc. The GADS system described above was written in FORTRAN, using an experimental graphical editor as the primary dialogue handling software and a laboratory enhanced raster scan color monitor.

The relationship between the three levels from which to view DSS is illustrated in Figure 2. The basic tools can be used to develop a specific DSS application directly. This is obviously the same approach used to develop most traditional applications, using tools such as a general purpose language, data access software, printer or terminal, etc. The constant change and flexibility of DSS make this approach difficult and costly. A serious complicating factor is the need to involve the user directly in the change and modification of the system. APL was heavily used in the development of early DSS because it proved cheap and easy for APL programmers (especially the APL enthusiasts) to produce "throw-away" codes which could be revised or discarded as the nature of the application changed. This language did not help capture the involvement of users in the building and modification of DSS, except for the few users who became members of the APL fan club.

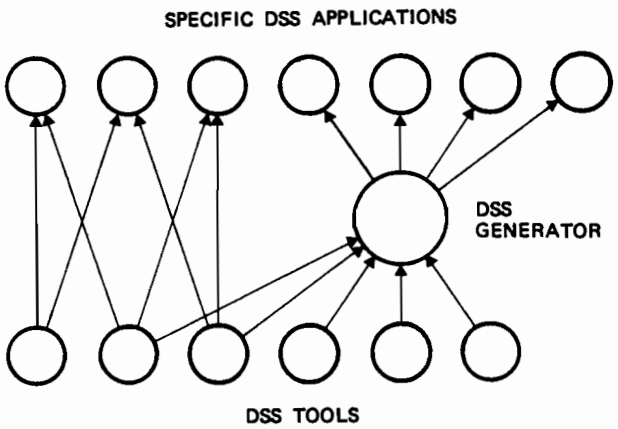


FIGURE 2 Three levels of DSS.

DSS generators have the objectives of shortening the development time for creating specific DSS applications and of encouraging substantive user participation in the process. In some cases, enlightened users may work directly with the generator to build a battery of applications to serve their information analysis and decision-making needs. In most cases, however, given the current state of technology and the attitudes of many managers, most users will need (or want) a "facilitator," who has a good knowledge of the problem area as well as some facility with and understanding of computer-based systems capabilities.

The role of the DSS generator, as used by the facilitator (sometimes called "builder"), is crucial to the concept of adaptive design for DSS. Because they must be continually flexible and adaptable to organizational and environmental changes, DSS are not amenable to the traditional system development process. Designers literally "cannot get to first base" because no one, least of all the decision maker or user, can define in advance what the functional requirements of the system should be. DSS developers must use the adaptive approach described by terms like permanent breadboarding, iterative design, or evolutionary development. In the following paper Peter Keen deals with the concept of adaptive design in some detail.

The three levels of hardware/software technology described above correspond to the viewpoints of three major "stakeholders," or interested parties, in the development and use of DSS. We can use these three levels to identify the characteristics and attributes of DSS as they are viewed by these three groups. At the top level are the *managers or users* who are concerned with what the DSS can do for them. Their concern is primarily with the problem-solving or decision-making tasks which they face in their organizational environment. They will assess the DSS in terms of their performance objectives in carrying out these tasks.

At the level of the DSS generator are the *builders* or *designers* who must use the capabilities of the generator to configure

a specific DSS to meet a given manager's needs. They will be concerned with the performance of the generator in terms of the capabilities it offers and how these capabilities can be assembled to create a specific DSS.

At the DSS tool level are the *technicians* or *"toolsmiths"* who build the generators from basic technology components. This group is concerned with basic tool development and the integration of these tools to form a DSS generator with the required capabilities.

Let us now look more closely at the attributes and characteristics of the DSS as viewed from each level. From the manager's view, we can identify six major performance objectives. As a group these represent the overall performance that seems to be expected and desirable from a managerial viewpoint (although other performance objectives could of course be considered). From the viewpoint of the designer, the characteristics of the DSS are described by a conceptual model which identifies three categories of performance characteristics: dialogue handling or the man-machine interface; data base and data base management capability; modeling and analytic capability. The sphere of interest of the "toolsmiths" may be described by the same three-part model; however, they focus specifically on the technology, tactics, and architecture needed to produce the capabilities required by the designers.

THE MANAGER'S VIEW: PERFORMANCE OBJECTIVES

The following performance requirements are phrased using the normative word "should." It is likely that no individual DSS will be required to satisfy all six of the performance requirements given here, but collectively they represent a set of capabilities which characterize the full value of the DSS concept.

(1) A DSS should provide support for decision making, with emphasis on semistructured and unstructured decisions. These are the types of decisions that have had little or no support from EDP, MIS, or management science/operations research (MS/OR) in the past. It might be better to refer to *hard* or *tough* problems, because the concept of "structure" in decision making is heavily dependent on the decision maker, his/her cognitive style, and his/her approach to problem solving. It is clear from their expressed concerns, however, that managers need additional support for certain kinds of problems.

(2) A DSS should provide decision-making support for managers at all levels, assisting in coordination between levels whenever appropriate. This requirement has evolved from the realization that managers at all organizational levels face "tough" problems (as described above). Moreover, a major need which they have articulated is the integration and coordination of decision making by managers dealing with related parts of a larger problem. We will have more to say about this objective in a moment.

(3) A DSS should support all phases of the decision-making process. A popular model of the decision-making process has been

given by Herbert Simon. He characterized the process in terms of three main steps:

— *Intelligence* (searching the environment for conditions calling for decisions). Raw data is obtained, processed, and examined for clues that may identify problems.
— *Design* (inventing, developing, and analyzing possible courses of action). This involves processes to understand the problem, to generate solutions, and to test solutions for reasonableness.
— *Choice* (selecting a particular course of action from those available). A choice is made and implemented.

Although the third step includes implementation, many authors feel that implementation is significant enough to be shown separately. It has been added to Figure 3, in order to show the relationships between all the steps.

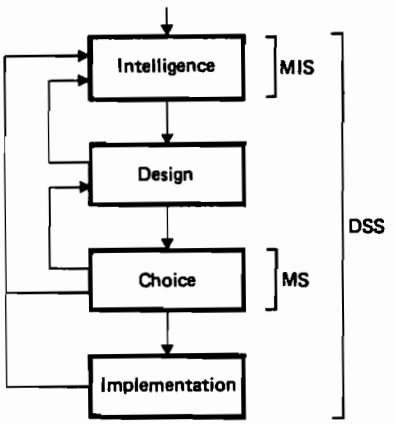


FIGURE 3 Phases of decision making.

Simon's model also illustrates the role of MIS and MS/OR in decision making. From the definition of the three steps given above, it is clear that EDP and MIS (in the narrow sense) have made major contributions to the intelligence phase, while MS/OR has been primarily useful at the choice phase. Thus far the design phase has not received substantial support from such information systems; this could be one of the primary potential contributions of DSS. There has also been very little support from traditional systems for the implementation phase; some early experience has shown that DSS can make a major contribution here.

(4) A DSS should support a variety of decision making processes, without being dependent on any one. Simon's model, though widely accepted, is only one model of how decisions are actually made. In fact, there is no universally accepted model of the decision-making process, and there is no promise of such

a general theory in the foreseeable future. There are just too many variables, too many different types of decisions, and too much variety in the characteristics of decision makers. Consequently, a very important characteristic of a DSS is that it does not impose a decision process on the user. It should provide the decision maker with a set of capabilities to apply in a sequence and form that fits his/her cognitive style. In short, a DSS should be process-independent and user-driven (or user-controlled).

(5) A DSS should support decisions which are *interdependent*, as well as those that are *independent*. Much of the early DSS work assumed that a decision maker would sit at a terminal, use a system, and develop a decision alone. DSS development experience has shown that a DSS must accommodate decisions which are made by groups or made in parts by several people in sequence. In this context Keen and Hackathorn (1979) have identified three types of decisions:

— *Independent*. A decision maker has full responsibility and authority to make a complete decision capable of implementation.
— *Sequential interdependent*. A decision maker makes part of a decision, which is then passed on to someone else.
— *Pooled interdependent*. A decision must result from negotiation and interaction between decision makers.

Different capabilities will be required to support each type of decision, i.e., personal support for independent decisions, organizational support for sequential interdependent decisions, and group support for pooled interdependent decisions.

(6) Finally, a DSS should be easy to use. A variety of terms have been used to describe this characteristic, including 'flexible,' 'user-friendly,' 'non-threatening,' etc. The importance of this characteristic is underscored by the discretionary latitude of a DSS's clientele. Although some systems with heavy organizational support or group support may limit the degree of discretion somewhat, the user of a DSS has much more latitude to ignore or circumvent the system than the user of a more traditional transaction system or a required reporting system. Therefore, the DSS must "earn" its users' allegiance by being valuable and convenient.

THE DESIGNER'S OR BUILDER'S VIEW

The designer (or builder or "facilitator") has the responsibility of drawing on computer-based tools and techniques to provide the decision support required by a manager. Basic tools can be used directly to develop a specific application, but as described earlier it is more efficient and effective to use a DSS generator for this task. The designer will require a set of capabilities which enable him to quickly and easily "configure" a specific DSS and to modify that DSS in response to changes in the manager's requirements, environment, tasks, and thinking approaches.

Let us consider a conceptual "black box" model which can organize these capabilities for the designers (as well as for the "toolsmith" who will develop the technology which provides these capabilities). As a preliminary step, recall that the true focus of attention is the Decision-making System, which consists of a *user/decision maker* faced with a *task* of some sort, in an *organizational environment*, using a set of capabilities which we call a DSS (see Figure 4). The conceptual model depicts the

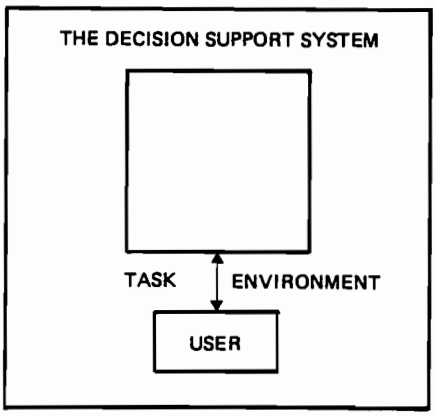


FIGURE 4 The decision-making system.

DSS system as a large black box containing smaller black boxes (subsystems). Upon opening the large DSS box (see Figure 5) we find that the system must have a data base and a model base, as well as a complex software system for linking the user to each of these. Opening each of these smaller boxes reveals that the data base and model base have some interrelated components and that the software system is comprised of three sets of capabilities: data base management software (DBMS); model base management software (MBMS); and software for managing the interface between the user and the system, which might be called the dialogue generation and management software (DGMS). The user-system interface, the data subsystem, and the model subsystem have become the framework for identifying the technical capabilities which a DSS must have. Let us consider the aspects of each of these subsystems that are critical to DSS.

The User-System Interface

The user-system interface consists of the user, the terminal, and the software for handling the user-system dialogue. The dialogue experience itself can be broken down into three parts (see Figure 6): what the user *sees* (the display or presentation language); what the user *can do* (the action language); and what the user *must know* (the user knowledge base). Bennett's work offers an excellent discussion of the specific DSS aspects of each of these areas and how they interrelate (Bennett 1976).

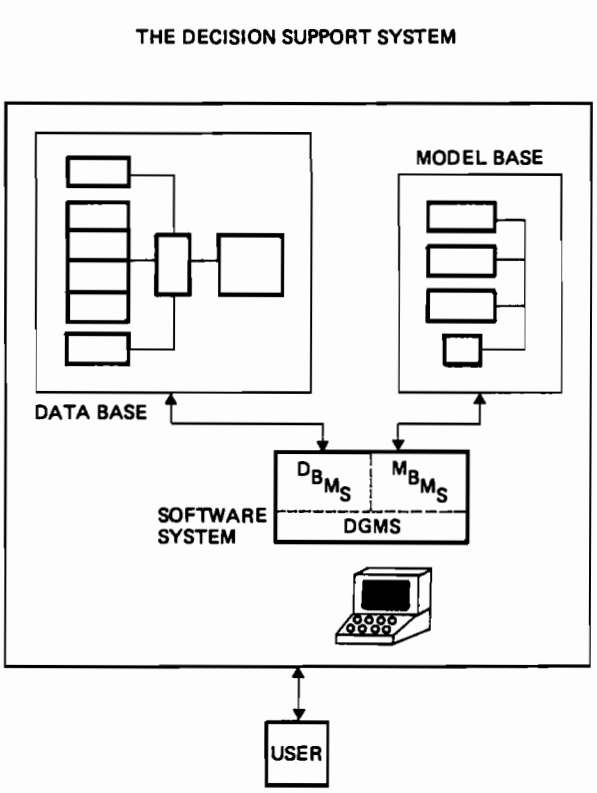


FIGURE 5 A conceptual model of a decision support system.

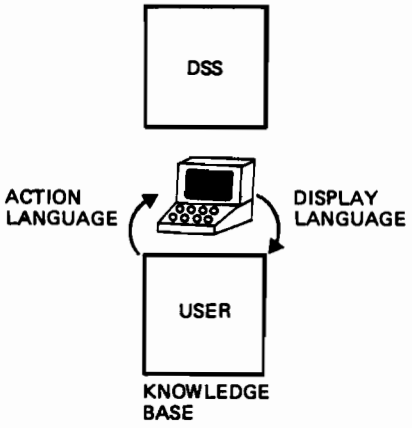


FIGURE 6 Elements of the user-system interface.

Carlson bases many of his design strategies on these three elements when he discusses representations (display), operation and control mechanisms (the action language), and memories (to support the knowledge base) (Carlson 1979). It is clear that a desirable set of capabilities includes

— the ability to handle a variety of dialogue styles;
— the ability to shift among them according to the user's choice;
— the ability to accommodate user actions in a variety of media;
— the ability to present data in a variety of formats and media;
— the ability to provide flexible support for the users' knowledge base.

The Data Subsystem

Because of the rapidly maturing technology related to data bases and their management, the data subsystem is considered a well-understood set of capabilities. The typical advantages of the data base approach and the powerful functions of the DBMS are important to the development and use of DSS. There are, however, some significant differences between the data base/data communication (DB/DC) approach for traditional systems and for DSS. First, DSS uses a much richer set of data sources than is usually found in typical nonDSS applications. Data must come from external as well as internal sources and from nontransactional sources. Decision making, especially in the upper-management levels, is heavily dependent on external data sources. At all levels, the typical accounting-oriented transaction data must be supplemented with nontransactional, nonaccounting data.

Another significant difference is the importance for DSS of the process of data capture and extraction from this wider set of data sources. Most successful DSS have found it necessary to create a DSS data base which is logically separate from other operational data bases. The nature of DSS requires that the extraction process and the DBMS that manages it be flexible enough to allow rapid additions and changes, in response to unanticipated user requests. A partial set of capabilities required in the data-base area can be summarized as follows:

— ability to combine a variety of data sources through a data capture and extraction process;
— ability to add and delete data sources quickly and easily;
— ability to portray logical data structures in user terms;
— ability to handle personal and unofficial data at the user's request;
— ability to manage this wide variety of data with a full range of data management functions.

The Model Subsystem

A promising aspect of DSS is their ability to make decision models available in a usable form to decision makers. Reasons

for the misuse and disuse of models have been widely discussed. One major problem has been the frequent preoccupation of model builders with model structure—the existence of the correct input data and the proper delivery of the output to the user being assumed. In addition to these heroic assumptions, models tended to suffer from inadequacy because of the difficulty of developing an integrated model to handle a realistic set of interrelated decisions. The solution tended to be a collection of separate models, each of which dealt with a distinct part of the problem. Communication between these related models was left to the decision maker as a manual and intellectual process.

A more enlightened view of models suggests that they be imbedded in an information system, with the data base acting as an integration and communication mechanism. The model creation process must be flexible, using a strong modeling language and a set of building blocks, much like subroutines, which can be assembled to assist the modeling process. In fact, there are a set of model management functions, analogous to data management functions. The key capabilities required for DSS in the modeling area include:

— ability to catalog and maintain a wide range of models, supporting all levels of management;
— ability to create new models quickly and easily;
— ability to interrelate these models with appropriate linkages through the data base;
— ability to manage the model base with software management functions analogous to data base management functions (such as a language for easy creation and modification of models, and a mechanism for storing, cataloging, linking, and accessing models).

THE TOOLSMITH'S VIEW

The toolsmith is concerned with creating the basic information technology and the architecture needed to combine the basic tools into a coherent system. We can use the same three-part model discussed in the section on the user-system interface to describe the toolsmith's concerns, because the tools must be designed and combined to provide the same three sets of capabilities, i.e., dialogue, data handling, and model building.

Each of these three areas has received a fair amount of attention from toolsmiths in the past. The foci of DSS and the requirements imposed by DSS have put these capabilities into a new light, revealing how they can be interrelated to increase their collective effectiveness. Moreover, the DSS requirements have revealed some missing elements in existing efforts, indicating important potential areas for development.

For example, there has been much theoretical and some empirical work on systems requirements for a good man-machine interface. Many studies have been based on watching users' behavior at terminals, or surveying users or programmers to ascertain what they

want in interactive systems. A recent study examined a series of interactive applications (many of which were DSS) to assess the *type* of software capabilities required by the applications (Sutton and Sprague 1978). This study has led directly to creative work on software architecture for dialogue generation and management systems (DGMS), as discussed in the previous section (Carlson and Metz 1980). For instance, a relation is employed as a data structure to store each picture or "frame" used in the system, and a decision table is employed to store the control mechanism for representing the potential users' option in branching from one frame to another.

There has been a significant amount of work in the data base management area during the past several years. Most has been aimed at transaction processing with large data bases. Most large DBMS have inquiry retrieval and flexible report preparation capabilities, but their largest contribution has been in the reduction of program maintenance costs through the separation of application programs and data definitions. It is significant that DBMS work has generally shown a rather naive view of the user and his requirements. A DSS user will not be satisfied merely with the capability to issue a set of retrieval commands that select items from the data base, or even to display those selected items in a report with flexible definition of format and headings. A DSS user needs to interact repeatedly and creatively with a relatively small set of data. He may only need 40 to 100 data variables, but they must be the *right* ones; these will change from day to day and week to week, and will include time series data which are not handled comprehensively by typical DBMS.

In short, there are some significant gaps in the development of DBMS when assessed from the viewpoint of DSS requirements. The critical area of data extraction with fast response, allowing additions and deletions to the DSS data base from the large transaction data base, was a major contribution of the GADS work (Carlson *et al.* 1974, Mantley and Carlson 1979). We will need better ways to handle and coordinate time series data as well as mechanisms for capturing, processing, and tagging judgmental and probabilistic data. In short, significant developments in the data-base area should be focused and extended in order to directly serve the needs of DSS.

The area of model creation and handling may have the greatest potential for contributing to DSS. The analytic capability provided by information systems has developed from statistical or financial analysis subroutines which can be called from a common command language. Modeling languages provide a way of formulating interrelationships between variables in a way that permits the creation of simulation or "what if" models. As we noted earlier, many of the currently viable DSS generators have evolved from these efforts. Early forms of "model management" seem to be evolving from enhancements to some modeling languages that permit simulation models to be used for sensitivity testing or goal seeking by specifying target and flexibility variables.

The model management area also has the potential for bringing some of the contributions of artificial intelligence to bear on

DSS. There have been a few tentative explorations of this idea (Bonezek *et al.* 1980), but "knowledge representation"—as a way of storing models linked with data—is as yet an untapped area.

RESEARCH

Let us now conclude by considering future research in DSS. The objective of this task force is to generate a clearer view of the full dimensions of DSS, and to develop some appropriate research directions. This paper attempted to present a framework for discussion and to indicate areas in which research activity is particularly needed. The three views of DSS described above and the associated levels of technology constitute three areas for research. One may thus consider the critical questions, the contributing or resource disciplines, and the research methodology which are appropriate at each level.

At this early stage of development in DSS, it is important that the word "research" be interpreted rather broadly. Indeed, to a large extent, the field of DSS rests on application experience. There is no orthodox body of theory or research methodology. Therefore, case studies, conceptual modeling, and actual development experience play a much stronger role than they might otherwise.

Possible research methodologies for DSS could be outlined in the following way:

(1) Undocumented wisdom based on experience; reactions from thoughtful observers and/or participants
(2) Surveys
(3) Case studies
    - Descriptive
        Unstructured/free format
        Structured so that several studies can be compared
    - Time dimension
        Static, snapshot, point-in-time
        Tracked over time
    - Author or case writer
        Independent observer
        Participant playing action/intervention role
(4) Laboratory research
    - Experimental lab with human subjects
        Real managers
        Surrogates, e.g., students as managers
    - Simulation lab
(5) Development research
(6) Model building
    - Conceptual
    - Theoretical

Table 1 is a partial list of the questions, disciplines, and methodologies which might be appropriate from the manager, designer, and toolsmith vantage points. It is not meant to be a comprehensive list, but rather a starting point for ideas. Each item

Table 1 A framework for Research in Decision Support Systems.

	Key questions	Foundation disciplines	Research methodologies
Managers	What do managers do? How do they make decisions? What activities are amenable to information systems support? What kinds of support are required?	Cognitive psychology Organization behavior and design Operations research/ management science	Case studies Experiments Normative theory development Decision research
Designers/ builders	What capabilities are required? How can the effectiveness of these capabilities be measured? What are the dynamics of the adaptive design approach (roles, processes, strategies) What are the relationships between systems (EDP, MIS, OA, etc.)?	MIS theory and practice Operations research/ management science Systems theory	Case studies Development experience Surveys
Toolsmith	What algorithms, storage structures, hardware are required? What is the "architecture" of the system? What are the modularity and interfacing requirements of the prototyping process?	Computer science Artificial intelligence	Theory development Development experience

in the table can be further specified by focusing on one of the managerial activities or one of the areas of technical capabilities.

SUMMARY

It is clear that DSS is evolving rapidly, driven by expressed needs from managers and organizations for more support from information technology, as well as by the rapid development of that technology. Full development of this field will require a creative coalescence of several foundation disciplines in order to bring information technology to bear on the problems in an effective way. Research opportunities and challenges are immense, and the potential rewards are correspondingly great.

REFERENCES

Bennett, John (1976) Integrating Users and Decision Support Systems. Proceedings of the Sixth and Seventh Annual Conferences of the Society for Management Information Systems, University of Michigan.

Bonezek, Robert H., Clyde W. Holsapple, and Andrew Whinston (1980) Evolving Roles of Models in Decision Support Systems. Decision Sciences 11(2).

Carlson, Eric D., J. Bennett, G. Giddings, and P. Mantey (1974) The Design and Evaluation of an Interactive Geo-Data Analysis and Display System. IFIP Congress 74, 1057-1061.

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

Carlson, Eric D. (1979) An Approach for Designing Decision Support Systems. Data Base 10(3): 3-15.

Carlson, Eric D., and Wolfgang Metz (1980) Integrating Dialog Management and Data Management. San Jose, California: IBM Research Division. IBM Research Report RJ2738.

Gerrity, Thomas P., Jr. (1971) Design of Man-Machine Decision Systems: An Application to Portfolio Management. Sloan Management Review 12(3): 59-75.

Keen, Peter G.W., and Michael S. Scott Morton (1978) Decision Support Systems: An Organizational Perspective. Reading, Massachusetts: Addison-Wesley Publishing Company.

Keen, Peter G.W., and Richard D. Hackathorn (1979) Decision Support Systems and Personal Computing. University of Pennsylvania, The Wharton School, Department of Decision Sciences. Working Paper 79-01-03.

Mantey, P.E., and Eric D. Carlson (1979) Integrated Geographic Data Bases: The GADS Experience. San Jose, California: IBM Research Division. IBM Research Report RJ2702.

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

Sutton, Jimmy A., and Ralph H. Sprague, Jr. (1978) A Study of Display Generation and Management in Interactive Business Applications. San Jose, California: IBM Research Division. IBM Research Report RJ2392.

Sutton, Jimmy A. (1978) Evaluation of a Decision Support System: A Case Study with the Office Products Division of IBM. San Jose, California: IBM Research Division. IBM Research Report RJ2214.

Zolliker, Mitch L., ed. (1980) Proceedings of a Conference on Application Development Systems. Data Base 11(3).

DECISION SUPPORT SYSTEMS: A RESEARCH PERSPECTIVE*

Peter G.W. Keen
Massachusetts Institute of Technology

INTRODUCTION

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

Research on DSS gained momentum around 1974. Only within the last two years has it reached a critical mass and expanded beyond a fairly narrow circle. By 1979 almost 30 fairly detailed case studies of DSS had been published. As the concept has become fashionable it has been used in looser and looser ways. Last year's article on "interactive marketing models" is cut and pasted and is resubmitted with "decision support systems" inserted instead. DSS may well be more important as a liberating rallying cry than

---

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

as a theoretical concept. However, the published case studies and conceptual proposals imply a coherent framework that makes DSS a meaningful discipline for both research and practice.

This paper presents a formal definition of DSS. It aims at answering two key questions: Is the term really necessary? If so, what are the research issues it implies? Its key argument is that the term DSS is relevant to situations where a "final" system can be developed only through an adaptive process of learning and evolution. The design strategy must then focus on completion; this is very different from the Management Science and Data Processing approaches. The research issues for DSS center around adaptation and evolution; they include managerial learning, representation of tasks and user behavior, design architecture, and strategies for beginning.

DEFINITIONS OF DSS

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

— A DSS is defined in terms of the *structure of the task* it addresses.
— DSS require a distinctive *design strategy* based on evolution and "middle-out" techniques.
— DSS support the cognitive processes of individual decision makers; *decision research* provides descriptive insights into management problem solving and normative theories for defining how to improve the effectiveness of such problem solving.
— DSS reflect an *implementation strategy* for making computers useful to managers; this strategy is based on the use of skilled intermediates, responsive service, and "humanized" software interfaces.

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

A potential strength of the DSS movement has been that it has at least tried to link theory and practice. It deals with real systems used in real organizations by real problem solvers, not with experiments involving captive students. At the same time, because it explicitly argues that DSS are different from

traditional systems, the better empirical work addresses conceptual issues, if only assertively. Available studies of DSS thus often provide illustrations, extensions, or counterexamples that can be used to test and extend their authors' conceptual assumptions.

CASE-BASED STUDIES OF DSS

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

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

(2) Usage is personalized; whether a system is recently operational or has been in place for some time, individuals' use of its functions varies widely.

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

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

(5) While the orthodox (academic) faith views DSS as tools for individual decision makers,** users regard the concept as more relevant to systems that support organizational processes. Users also feel that they do not actually use DSS for decision making.

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

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

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

---

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

**Keen and Morton's (1978) book on DSS is mistakenly subtitled "An Organizational Perspective"; the authors herewith recant.

— Personal Support Systems (PSS), for use by individuals in tasks that do not involve interdependencies, so that the user can indeed make a decision;
— Group Support Systems (GSS), for tasks with "pooled" interdependencies, which thus require substantial face-to-face discussion and communication; and
— Organizational Support Systems (OSS), for tasks that involve "sequential" interdependencies.

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

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

MIDDLE-OUT DESIGN

The studies strongly support the concept of middle-out design for DSS.* Most descriptions of DSS implementation highlight careful use of prototypes, continued incremental development, and response to users' changing demands. Writers such as Ness (1975); Courbon, Grajew, and Tolovi; and Berger and Edelman (1977) make a strong implicit case for viewing X Support Systems (where X may stand for Decision, Management, Personal, Organizational, Interactive, or some other modifier) as an adaptive design strategy.

The obvious question is whether the strategy is general for interactive systems or needed only for particular situations. Middle-out design differs most from traditional techniques in that it explicitly proceeds without functional specifications. Data Processing (DP) has shown, through vicarious trial-and-error learning and occasional reflection, that systems development requires planning before programming. Brooks' brilliant and somewhat rueful review of software engineering, *The Mythical Man-Month* (1975), established that coding is only 10% of the total effort in the system's development life cycle. Standard textbooks generally recommend that approximately 40% of the effort go to analysis and specifications, 10% to coding, 30% to testing, and 20% to installation (and another 100%-300% to maintenance). The

---

*The term was created by Ness, who built many of the early DSS and trained, at Massachusetts Institute of Technology (MIT) and Wharton, many DSS designers. His working papers and case studies in Alter (1980) and in Keen and Morton (1978) show the development of the middle-out concept. Courbon, Grajew, and Tolovi have extended it in some brilliant empirical studies; they use the term "l'approche evolutive."

vocabulary of DP is full of terms like "signing-off," "functional specifications," and making a system "operational."

DSS case studies, including those that were not based on middle-out design strategy, contradict the recommendations underlying the systems development life cycle. This clearly implies that defining a system as a DSS, rather than, possibly, an interactive information retrieval system, *does* make a difference. It shifts the focus of the development process from delaying coding to starting on it as quickly as possible—from aiming toward a clearly defined "final" system to implementing an initial one that can then be firmed up, modified, and evolved. The systems development life cycle is a strategy for finishing; adaptive design (a term that captures all the middle-out, incremental, and evolutionary techniques scattered throughout the case studies) is a method for beginning.

"SEMISTRUCTURED" TASKS

Viewing DSS in terms of the design process is not enough to integrate all the conclusions from the case studies. It also sidesteps key conceptual issues raised by the decision research and task-centered conceptions of DSS. Gorry and Morton's paper "A Framework for Management Information Systems" (1971) was a seminal work for DSS. It built on Simon's concept of programmed and nonprogrammed tasks, identifying "semistructured" tasks as those requiring special treatment. Structured tasks can be automated or routinized, thus replacing judgment, while unstructured ones involve judgment entirely and defy computerization. Semistructured tasks permit a synthesis of human judgment and the computer's capabilities.

There are several problems with this argument. The terms "structured" and "unstructured" point to a spectrum of tasks, but there is no actual operationalization of "semistructured." More importantly, it is unclear whether structure is perceptual or intrinsic to the task. Stabell (1977) also points out that organizations often *decide* to treat an unstructured task as if it were structured; the degree of structure is then socially defined, as well as perceptual.

The Gorry-Morton framework is not a complete or convincing theoretical statement; the range of applications, technologies, and mode of use of the DSS described in the case studies is too broad to fit into it. (This applies also to Gorry and Morton's use of Anthony's distinction between strategic planning, management control, and operational control. Morton (1971) suggests that DSS apply to the first two areas, but Berger and Edelman (1977) give striking examples of a DSS for operational control.)

Despite the looseness of its definition and the lack of comprehensive supporting evidence in the case studies, Gorry and Morton's notion of semistructured tasks is intuitively convincing. Keen and Morton rely on it in explaining the concept of support, rather than replacement, of managerial judgment. Any effort to

define how a DSS helps improve effectiveness in decision making, and not just efficiency, has to introduce some similar notion of the relationship between task structure and process (Stabell 1977, Carlson and Sutton 1974).

DSS REDEFINED

A central argument of this paper is that what Gorry and Morton (1971) present and Gerrity (1970), Morton (1971), Stabell (1977), and Keen and Morton (1978) extend, is not the general case but a special one. The definition of Support Systems that follows combines the task-centered perspective with that of adaptive design and also picks up on the most interesting finding from the case studies, the unpredictability of DSS usage. *The label "Support System" is meaningful only in situations where the "final" system must emerge through an adaptive process of design and usage.*

This process may be needed for any of the following reasons:

(1) The designer or user cannot provide functional specifications or is unwilling to do so. A "semistructured" task is such an instance; we either lack the knowledge necessary to lay out procedures and requirements (i.e., the degree of structure is perceptual) or feel that such a statement can never be made (i.e., the lack of structure is intrinsic to the task).

(2) Users do not know what they want and designers do not understand what users need or can accept; an initial system must be built to give users something concrete to which to react. (This is the assumption underlying middle-out design.)

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

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

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

This view makes DSS a necessary concept. For any given system development effort, it is important whether or not the implementers view it as requiring a DSS rather than a marketing model, retrieval system, report generator, or other alternative. Reliance on traditional development techniques would be a severe mistake in a case where the final system will evolve only through

the ongoing interaction of designer and user, learning, personalized use, or the evolution of new functions. Learning, adaptation, and evolution are made feasible by building a "DSS" and not a "model." If these are not needed for effective development and use of a system, then one should build it as a "model" in the traditional way; the new label is not relevant.

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

— understanding the dynamics of the adaptive relationship between user, designer, and technical system;
— analyzing tasks in relation to users' processes and criteria for system design; and
— developing an organizational focus to complement the cognitive perspective and thus include Organizational as well as Personal Support Systems.

ADAPTIVE DEVELOPMENT AND USE

Figure 1 shows the adaptive links between the major actors involved in any DSS development and the technical system. The arrows represent a direction of influence. For example, SYSTEM (S) → USER(U) indicates that learning is stimulated by the DSS,

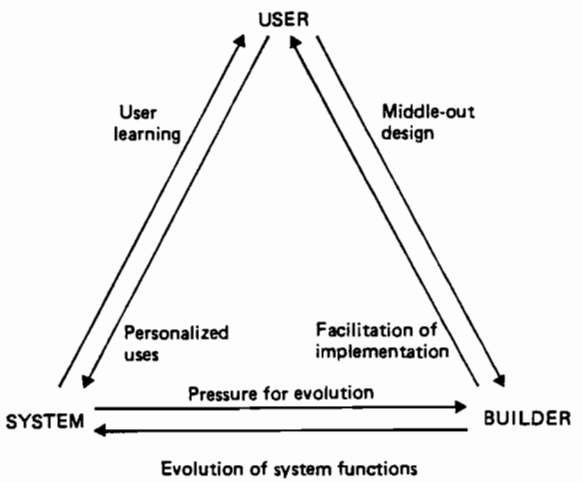
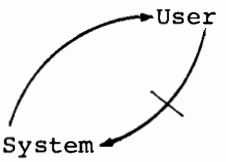


FIGURE 1 An adaptive framework for DSS.

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

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

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



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

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

— A system is a "DSS" only if each of the arrows is relevant to the situation.
— Where the arrows are relevant, the design process must ensure that they are not blocked by inflexible design structures, failure to allocate resources for implementing new functions, or lack of a direct relationship between user and designer.
— Each arrow represents a distinctive aspect of research and practice.

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

S⇄U, U⇄B, and S⇄B

The System-User Link

In the context of Personal Support Systems, S⇄U may be termed the *cognitive loop*. (The issue of organizational support will be discussed separately.) The link S→U concerns managerial learning and U→S individuals' exploitation of the DSS capabilities, their own learning, or both. The cognitive loop helps to explain the consistent finding in the case studies that individuals use a given DSS in different ways and that uses are thus often unintended and unpredicted. This seems a natural outcome of the sequence S→U→S→U ....

Much early DSS research explored aspects of the cognitive loop, particularly characteristics of individual problem solving that influence the use of a DSS. This was a fairly static analysis; examining the managerial (and organizational) learning process in more detail seems essential. Doing so requires richer theoretical models; early research drew on limited concepts of cognitive style and cognitive structure, which were at too high a level of analysis to track the learning process. They focused on general aspects of the psychology of individual differences (Stabell 1977, Carlisle 1974, Grochow 1974).

The User-Builder Link

The link U⇄B is the *implementation loop*. U→B highlights a key aspect of adaptive design discussed by Ness (1975) and Courbon *et al*. Ackoff (1960) long ago pointed out that users do not know what they need. The middle-out approach relies on the quick delivery of an initial system to which users can respond and thus clarify their desires. Middle-out design is the means by which the *designer* learns from the *user*; it also ensures that the user drives the design process.

B→U has been explored in studies of DSS implementation that examine the role of the "integrating agent" (Bennett 1976), intermediary (Keen 1976), chauffeur (Grace 1976), and change agent (Ginzberg 1975). DSS are a service rather than a product and require the designer to understand users' perspective and processes, to build credibility, and to be responsive to users' evolving needs.

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

The System-Builder Link

The *evolution loop* (S⇄B) is less easy to label than are the others. While the case studies repeatedly show that DSS evolve, and while much of the conceptual work relevant to DSS recommends evolutionary development, there are few detailed

longitudinal studies or theoretical models. It is perhaps easiest to view the links in relation to the other loops. Managerial learning (S⟶U) and personalized uses (U⟶S) put strain on the existing system. This builds pressure for evolution (S⟶B). New functions are then provided (B⟶S). The case studies imply that this is not a continued, evenly paced process, but that it occurs in discrete phases (see also Andreoli and Steadman 1975). Users explore the initial system and gradually become confident with it. At a certain point, it becomes apparent that a new function needs to be added to the system. Quite often, usage does not "take off" until this extension is provided; the "new" system leads to very different volumes and modes of use than did the earlier one (Andreoli and Steadman 1975).

The S⟶B link needs research. Keen and Gambino (1980) have employed the common device of a data trap to track individuals' use of a DSS* (see also Stabell 1977, Andreoli and Steadman 1975) in terms of emerging patterns and "command sequences." The argument is that users initially use DSS commands as single words (e.g., 'LIST', 'REGRESS') but later develop, largely via the adaptive processes of the cognitive loop, what are effectively sentences; they use consistent sequences of commands and build up their own analytic routines. This process is easy to identify; the hypothesis is that it triggers demand for or readiness to use new commands.

The other link, B⟶S, is easier to explain. It simply involves the designer adding new capabilities to the DSS. This obviously is feasible only if the design architecture is modular, flexible, and easily modified, if the programmer can implement new functions inexpensively and quickly, and if the designer maintains ongoing contact with users. The advocates of APL as "the" language for DSS, the advocates of end-user languages, and the advocates of "command-driven" interfaces all emphasize the need for program structures and programming methods to facilitate evolution. Case studies indicate that success of a DSS often depends on its evolution rather than its initial use, and on fast, responsive implementation.

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

This paper cannot treat each adaptive link in Figure 1 in any detail; the preceding discussion covers only a few issues relevant to research. In Figure 1, the author seeks to present a definition of DSS development that clarifies what a DSS is and

---

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

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

what it is not, as well as which actions and processes it involves. Each arrow represents a clear research area relevant to DSS practice.

THE TASK CONTEXT

Figure 1 ignores the task to be supported. Obviously, a DSS can be built only if the designer understands the task at a level of detail sufficient to relate the task to the users' processes, design the functions of the DSS, and by extension, relate the users' processes to the DSS functions.

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

The four ideas that follow require a major research effort** before they can be validated and made operational. In a way, they pick up on Gorry and Morton's discussion of semistructured tasks at a more molecular level.

- The tasks a DSS addresses involve some degree of discretion, inference, and selection of information; if this is not so, there are no adaptive links between user and system (U⟷S). A whole task is composed of subtasks. The whole task may be the university admissions decision, portfolio management, or media selection. The subtasks are discrete intellectual operations, such as *calculating* a sum, *searching* for a value, or *comparing* two variables on a graph.
- The subtasks identify the potential functions for the DSS, e.g., CALC, FIND, COMPARE.
- User behavior and user learning can be described in terms of the sequence of and change in subtasks.
- Use of the DSS can be tracked in relation to the functions.

This level of representation has several practical and conceptual merits. It also suggests that DSS should be command-driven. Keen and Alter argue that the commands correspond to the users' verbs (e.g., 'list', 'graph'). Keen adds that if a function in a system does not directly relate to some concept in the users' mind, it cannot be used. Carrying out a task

---

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

**Henderson, Gambino, and Ghani: private communication.

involves a sequence of verb-related subtasks (Do this ...; then this ...). Using a DSS involves invoking a sequence of verb-based commands. Evolving it means adding new commands. Learning is identifiable only in terms of using new commands or redefining existing ones, and personalized use is apparent from the choice of commands.

In a structured task, subtasks can be clearly specified and the sequence in which they are invoked can be predicted. A semi-structured task is thus one where either (1) not all the subtasks can be represented and hence translated into DSS commands or (2) all the subtasks can be represented but the sequence cannot be predicted. Focussing on subtasks rather than on whole tasks retains the intuitive appeal of the Gorry-Morton framework but eliminates its problems of definition. In addition, doing so addresses Stabell's point that a whole task is often socially defined; while two universities may handle the admissions process (the whole task) differently, it will have common subtasks.

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

A number of DSS researchers share this focus on subtasks. Blanning (1979) outlines the equivalent of a generative grammar for DSS that goes beyond verbs. Keen and Gambino (1980) suggest that most whole tasks require a common set of verbs; almost any DSS needs such functions as Graph, List, Select, and Describe (provide descriptive statistics). Henderson and his team designed a set of experiments on DSS use that track user behavior at the command and subtask level.*

These tentative ideas constitute a proposal for research rather than a conclusion from it. The central postulate is that adaptive design and use of DSS, DSS evolution, managerial learning, and so forth require a decision research process where the level of analysis is at the subtask level. Much of the vagueness of DSS concepts disappears when this is provided. Of course, the research issue is how to represent the subtasks. Contreras (1978), following on Berry (1977), argues that subtasks are linguistically at the level of APL functions, which can be further broken down into primitives. Blanning (1979) adopts a similar perspective.

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

---

*Henderson, Gambino, and Ghani: private communication.

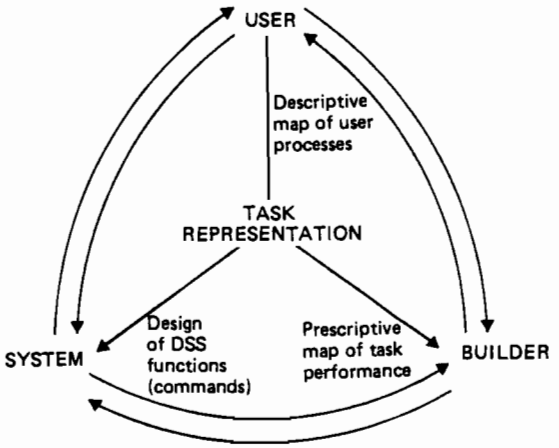


FIGURE 2 Task context.

CONTEXTUAL ISSUES IN DSS DEVELOPMENT

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

In a similar fashion, the DSS itself is constrained by the organization's available technology (T$\longrightarrow$S). This includes data as well as computer power and the base of reliable operational systems and technical expertise on which a DSS capability is built. While the case studies generally describe successful systems, several suggest that DSS will not take root in an organization that has not yet provided managers with standard data processing and reporting systems. In such situations, DSS are seen as a luxury or as irrelevant.

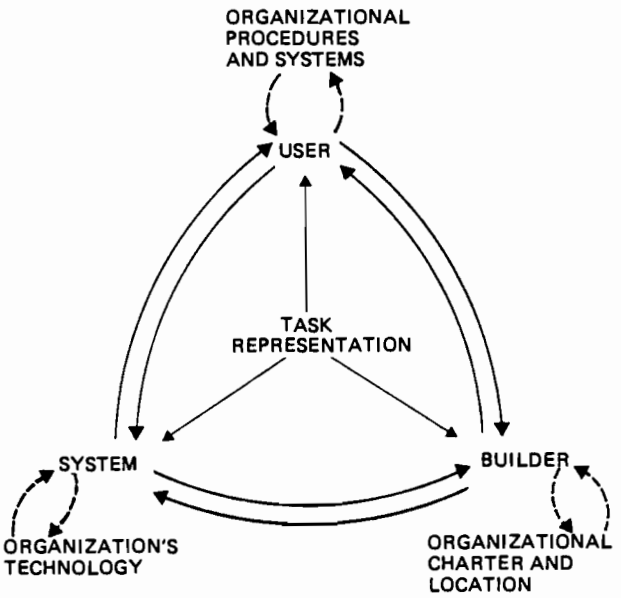


FIGURE 3 Organizational issues.

The link S- - - -T is a reminder that learning and evolution can be blocked by the inability to obtain additional technology. Keen and Clark (1979) found that use of their DSS for state government policy and analysis was strongly constrained by states' existing technology and procedures for operating it. In addition, managerial learning and evolution of the DSS may require new data and structures or lead to an overloading of the organization's time-sharing computer. The whole adaptive process in Figure 3 breaks down when any influence is absent or blocked.

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

— the two groups are geographically or psychologically isolated;
— the designers are part of a unit, such as Data Processing, with no mandate for innovation; or
— the organization's charge-out policies and procedures for project justification discourage exploration and require "hard" benefits. Keen points out that DSS often provide primarily

qualitative benefits. They "improve" a decision process and it is unlikely that one can point in advance to a "bottom-line" payoff, especially if the value of the system is in the end determined by an adaptive, evolutionary process.

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

CONCLUSION

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

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

APPENDIX 1: CASE STUDIES OF DSS

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

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

(2) Carlson, editor (1977) *Proceedings of a Conference on Decision Support Systems* contains very little conceptual material and emphasizes real-world applications. It has a strong "show-and-tell" flavor. Whereas Keen and Morton use case descriptions to illustrate the concepts of a DSS, practitioners in this volume demonstrate their view of what aspects of the concepts have practical value. On the whole, they do not share Keen and Morton's emphasis on the cognitive characteristics of the individual decision maker but instead focus on organizational processes.

(3) Alter (1979) *Decision Support Systems: Current Practice and Continuing Challenges* is based on case studies of 56 systems, only a few of which are DSS. There are 7 detailed cases, some of which overlap with Keen and Morton and with Carlson. Alter is in part concerned with sharpening the practical definitions of DSS by looking at innovative systems in general. He uses the term DSS fairly loosely, primarily because his is an exploratory study that asks specifically whether it is useful to identify a system as a DSS.

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

The best bibliographies on DSS are in Keen and Morton and in Grajew and Tolovi. In the list that follows, the major sources of reference are identified. The major cases are:

AAIMS:	An Analytic Information Management System (Carlson, Alter)
BIS:	Budget Information System (Alter)
BRANDAID:	Marketing Brand Management (Keen and Morton)
CAUSE:	Computer Assisted Underwriting System at Equitable (Alter)
CIS:	Capacity Information System (Keen and Morton)
EIS:	Executive Information System (Carlson)
GADS:	Geodata Analysis Display System (Keen and Morton)
GMIS:	Generalized Management Information System (Keen and Morton)
GPLAN:	Generalized Planning (Carlson)
IMS:	Interactive Marketing System (Alter)
IRIS:	Industrial Relations Information System (Carlson)
ISSPA:	Interactive Support System for Policy Analysts (Keen and Gambino)
MAPP:	Managerial Analysis for Profit Planning (Carlson)
PDSS:	Procurement Decision Support System (International Harvester, private paper)
PMS:	Portfolio Management System (Keen and Morton, Alter)
PROJECTOR:	Strategic Financial Planning (Keen and Morton)
REGIS:	Relational Generalized Information System (Carlson)

APPENDIX 2: SOME CHARACTERISTICS OF SELECTED DSS

Unanticipated Uses

— PMS. Intended use: investment decision tool. Actual uses: marketing tool and customer relations aid.
— MAPP. Intended use: financial planning. Actual use: revealing branch bank irregularities.
— PROJECTOR. Intended use: analyzing financial data to answer preplanned questions. Actual use: alerting users to new issues and unanticipated questions.

Personalized Uses

— GADS. Public officials (police and school system users) could imagine solutions and then use GADS to test hypotheses; individual users' values placed on variables led to entirely different conclusions.
— REGIS. REGIS encouraged data browsing, discerning new relationships and questions.
— PMS. Individual Managers used widely different function combinations.

Evolution

— BIS. Initial system modular in structure; data base separate from applications programs; new programs added incrementally without upsetting data base.
— PMS. Initial prototype followed by full implementation; number of programs doubled in six months.
— CAUSE. Four evolutionary versions; deliberate emphasis on phased development to build credibility and capability: routines increased from 26 to 200 during the evolutionary period.

Simple Functions

— AIMS. DISPLAY, PLOT, QUARTERLY, CHANGE ... (60 verb-like commands used).
— ISSPA. DESCRIBE, EQUITY, REGRESS, HISTO, RANK, NTILES ....
— PMS. SCATTER, SCAN, STATUS, TABLE, GRAPH, SUMMARY, GROUP ....

Organizational Support System

— CAUSE. Supports underwriting process, including data definition and collection.
— PDSS. Stabilizes purchasing agents' ordering system.
— IRIS. Supports operations control in industrial relations applications.

Benefits

— CAUSE. Reduced need for employer specialization; increased possibilities of internal reorganization; gave opportunity to newer employees.
— PROJECTOR. Improved time effectiveness "by a factor of 20"; forced consideration of related issues; produced "confidence-inspiring" analysis.
— MAPP. Provided better product definitions and costing allocation; promoted internal learning.

Intermediaries

— GADS. Chauffeur used as teacher and translator to save time in finding as many solutions as possible as quickly as possible.
— IMS. Junior researcher with no decision-making authority used 50% of time; intermediary used only to push buttons, not to make decisions.
— PMS. Secretaries operate; managers specify desired output.

REFERENCES

Ackoff, R.L. (1960) Unsuccessful Case Studies and Why. Operations Research 8(4): 259-263.

Alter, S.L. (1979) Decision Support Systems: Current Practice and Continuing Challenges. Reading, Massachusetts: Addison-Wesley.

Andreoli, P., and J. Steadman (1975) Management Decision Support Systems: Impact on the Decision Process. Master's thesis, Massachusetts Institute of Technology.

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

Bennett, J. (1976) Integrating Users and Decision Support Systems. Pages 77-86, Proceedings of the Sixth and Seventh Annual Conferences of the Society for Management Information Systems, edited by J.D. White. Ann Arbor: University of Michigan.

Berger, P., and F. Edelman (1977) IRIS: A Transaction-Based Decision Support System for Human Resources Management. Database 8(3).

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

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

Brooks, F.P. (1975) The Mythical Man-Month. Reading, Massachusetts: Addison-Wesley.

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

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

Carlson, E.D., ed. (1977) Proceedings of a Conference on Decision Support Systems. Data Base 8(3).

Contreras, L. (1978) Decision Support Systems and Corporate Planning. Informal communication.

Courbon, J.C., J. Grajew, and J. Tolovi. Design and Implementation of Interactive Decision Support Systems: An Evolutive Approach. Grenoble, France: Institut d'Administration des Entreprises. Unpublished.

Gerrity, T.P., Jr. (1970) The Design of Man-Machine Decision Systems. Ph.D. dissertation, Massachusetts Institute of Technology.

Ginzberg, M.J. (1975) A Process Approach to Management Science Implementation. Ph.D. dissertation, Massachusetts Institute of Technology.

Gorry, G.A., and M.S. Scott Morton (1971) A Framework for Management Information Systems. Sloan Management Review 13(1): 55-70.

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

Grajew, J., and J. Tolovi (1978) Conception et mise en oeuvre des Systèmes Interactifs d'aide à la décision. Doctoral thesis, University of Grenoble.

Grochow, J.M. (1974) Cognitive Style as a Factor in the Use of Interactive Computer Systems for Decision Support. Cambridge, Massachusetts: Massachusetts Institute of Technology.

Hackathorn, R.D. (1978) Research Issues in Personal Computing. Washington, D.C.: Proceedings of the National ACM Conference.

Keen, P.G.W. (1976) Computer Systems for Top Managers: A Modest Proposal. Sloan Management Review 18(1): 1-17.

Keen, P.G.W., and Clark (1979) Simulations for School Finance: A Survey and Assessment. Research Report to Ford Foundation.

Keen, P.G.W., and T. Gambino (1980) Mythical Man-Month Revisited. Center for Information Systems Research working paper.

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

Keen, P.G.W., and G.R. Wagner (1979) Implementing Decision Support Systems Philosophy. Datamation 25(12): 117-122.

Meldman, J. (1977) Decision Support Systems for Legal Research. Monterrey, Mexico: Paper presented at Eleventh Symposium Nacional de Systemas Computacionales. March 15-18, 1977.

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

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

Simon, H.A. (1957) A Behavioral Model of Rational Choice. Pages 241-260, H.A. Simon, Models of Man. New York: Wiley.

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

ORGANIZATIONAL SCIENCE CONTRIBUTIONS TO THE DESIGN OF DECISION SUPPORT SYSTEMS

George P. Huber
University of Wisconsin

INTRODUCTION

Exceedingly few of the world's managers have access to a Decision Support System (a DSS, as defined by the books, articles, and marketing materials that use this term). On the other hand, every manager has a "decision support system" (a dss, consisting of the information sources and decision-aiding processes that he or she draws upon as the occasion requires.*

One of the issues that will undoubtedly be discussed at this Task Force Meeting, either formally or informally, is the relationship between Decision Support Systems (DSS) and decision support systems (dss). For example, we may at some point address the following two questions:

(1) Assuming that the interface between a DSS and the dss in which it is implanted should not be entirely rigid and impermeable, to what extent should we attempt to identify (or design) the connecting linkages?

(2) Recognizing that both the DSS and the dss are themselves supported by an "organizational information system"—a system that we know to be subject to a variety of malfunctions—what can we do to increase the effectiveness of this supporting system?

Obviously, no one discipline or perspective can provide complete answers to these questions or a complete knowledge base for the design of Decision Support Systems. This should be kept in mind during the following discussion of the contributions

---

*We will discuss some of the information requirements of such a system in a later section of this paper.

that organizational scientists can make to the design of Decision Support Systems. One contribution is to remind us of the fact that *DSSs that are compatible with the manager's dss will be more used and useful than those that are not.* A wealth of organizational science studies on organizational change and on technological innovation attest to this. It is an undeniable fact, but one that bears repeating.

A second contribution of organizational scientists stems from their knowledge and portrayal of organizational information systems. Through their studies, they have learned a great deal about the performance and behavior of such systems, and we would be wise to take account of what they know. An examination of the associated literature (Huber 1980a, 1980b) indicates that the results of these studies are applicable to the design of both computer-aided information systems and traditional information systems.

A third contribution of organizational scientists derives from their knowledge and portrayal of the nature of managerial decision processes in general, which makes possible better diagnoses of those situations appropriate for the design and implementation of Decision Support Systems. The following remarks focus on this contribution. We will discuss what organizational scientists have learned about the nature of managerial decision making and about the nature of the information that managers use to support their decision processes.

Keen and Morton remind us that "A main argument of the DSS approach is that effective design depends on the technician's detailed understanding of management decision processes" (Keen and Morton 1978, p. 1). With this view in mind, let us turn to a discussion of the nature of managerial decision making, or what may more accurately be called "organizational decision making."*

Organizational decision making is the process by which one or more organizational units make a decision on behalf of the organization. The decision-making units can be as small as one individual, e.g., a manager, or as large as the entire organizational membership. While personal goals often influence organizational decisions, the decisions themselves are legitimized to other units and agencies in terms of fulfilling organizational needs. As Barnard put the matter, "Organizational decisions do not relate to personal purposes, but to organization purposes" (Barnard 1938, p. 195).

Clearly a good deal of literature relates to this topic, including literature on topics as diverse as operations research, small group behavior, and legislative processes. In keeping with the objectives of the Task Force Meeting and in order to

---

*The change in terminology reminds us of two facts. One is that managerial decisions are always made within and influenced by an organizational setting. The other is that in many cases managers do not make decisions, but rather manage decision processes or cause decisions to be made.

narrow the scope of the paper, we have focused on decisions made on behalf of formal, complex organizations. For the same reasons, we have directed our attention to the organizational science literature—the literature that attempts to predict, explain, or interpret organizational decision processes and outcomes.

ORGANIZATIONAL DECISION MAKING

Four conceptual models of organizational decision making are described in this section, namely, the Rational Model, the Political Model, the Garbage Can Model, and the Program Model. We will begin with the Rational Model, since the other three models are invariably portrayed as relating to it in some way, e.g., as an alternative or as a complementary elaboration.

The Rational Model

The Rational Model suggests that *organizational decisions are consequences of organizational units using information in an intendedly rational manner to make choices on behalf of the organization (Proposition R)*. The qualifier "intendedly" acknowledges the fact that although decision makers may attempt to use normatively correct decision procedures, they are generally unsuccessful, due to either intellectual or resource constraints.

A number of bodies of literature are related to this model. One consists of reports on the use of normative procedures, such as operations research or program evaluation, in organizational decision making (see Howard *et al.* 1976, Kaplan and Schwartz 1977). Another consists of investigations of the apparently rational use of information by individuals in simulated organizational environments (see Mobley and Meglino 1977, Keen and Morton 1978). Most of the latter studies are attempts to test the applicability of behavioral decision theory (see Slovic *et al.* 1977) in organizational settings. A third body of literature consists of authoritative descriptions of the nature of managerial decision making (Simon 1947, March and Simon 1958, Downs 1966, Mintzberg *et al.* 1976). Although the Rational Model is an extremely important model, both because it is often publicly espoused and because it is often the target of those who prefer alternative models, it is easy to forget that it is simply a model, an abstraction of reality, a fragmentary representation of what actually occurs, and an inadequate basis for the design of the Decision Support Systems that we aspire to create.

The Political Model

The Political Model emphasizes that *organizational decisions are consequences of the application of strategies and tactics by units seeking to influence decision processes in directions that will result in choices favorable to themselves (Proposition POL)*.

A good deal of literature addresses the issue of organizational politics, although a composite model or theory is not yet available. Some of this literature concerns organizational power, particularly how such power is acquired (see Mechanic 1962, Pettigrew 1972). Another part of the literature describes field studies in which the decision processes and outcomes are interpreted as having political bases (Pfeffer *et al.* 1976, Hills and Mahoney 1978). Expositions on the role of politics in complex organizations are provided by March (1962) and Pettigrew (1973).

The literature on conflict and bargaining is also related to the literature on organization politics; however, with few exceptions (see Walton *et al.* 1969, Schmidt and Kochan 1972), it is not based on studies undertaken in the complex and hierarchical organizations in which we are primarily interested.

The Garbage Can Model

The Garbage Can Model is relatively new. Although the basic concept was alluded to in 1963 (Cyert and March 1963, pp. 80, 121), the model itself was not fully explicated until 1972 (Cohen *et al.* 1972). The essence of the model may be described as follows. *Organizational decisions are consequences of intersections of problems looking for solutions, solutions looking for problems, and opportunities for making decisions (Proposition GC).*

These three variables—problems, solutions, and choice opportunities—along with a fourth—participants—are portrayed by Cohen *et al.* (1972) and March and Olsen (1976) as being tossed into and churning about in a garbage can.

In spite of the superficiality suggested by its name and its apparent simplicity, the model is important. On one hand, it is apparently so useful in interpreting a wide variety of organizational decisions (see March and Olsen 1976), and yet, on the other hand, it relies so little on available behavioral or normative theories of decision making. To say the least, the model is thought-provoking (see Moch and Pondy 1977) or, as Perrow chooses to say, "aggravating" (Perrow 1979, p. 153).

To the author's knowledge, the model has not been empirically tested by anyone other than March and Olsen and their associates, though Ritti and Goldner observed some organizational behavior that the model would predict (Ritti and Goldner 1969, p. B-23E).

Although it is applied in an organizational context, the model seems to have wider relevance. For example, in the field of marketing (as contrasted with the fields of organizational behavior and theory), advertising is a well-known mechanism for making certain "solutions" conspicuous to potential consumers and thereby increasing the likelihood that these "participants" will choose the product or service when they have a "problem."

The Process Model

In 1888 James Bryce observed that "to the vast majority of mankind, nothing is more agreeable than to escape the need for mental exertion...to most people nothing is more troublesome than the effort of thinking" (Bryce 1888). Sixty years later Chester Barnard made a similar observation in his book *The Functions of the Executive* (Barnard 1938, p. 189):

> The making of decisions, as everyone knows from personal experience, is a burdensome task.... Accordingly, it will be observed that men generally try to avoid making decisions, beyond a limited degree when they are rather uncritical responses to conditions.

Herbert Simon won the Nobel prize in 1978 for his writings suggesting that organizational decision making was significantly and adversely affected by the cognitive limitations of the human decision maker (among other professional contributions) (see Simon 1947, March and Simon 1958). In 1966 Anthony Downs suggested that the nature of a manager's task environment was not conducive to high-quality decision making (Downs 1966, p. 75), a point of view that seems supported by Mintzberg's studies (see Mintzberg 1975).

This evidence, together with authoritative testimony, leads one to ask, "When rationality and analysis do not guide organizational decision making, what does?" The Process Model identifies two factors that seem to determine many organizational decisions. One is "programs." Our everyday observations and a good many empirical studies remind us that decision-making behavior in organizations is affected by standard procedures, group norms, budget limitations, and other forms of action-directing or action-constraining organizational "programs."

The other factor that seems to guide decision making in a wide variety of circumstances is "programming." Again our own observations and an abundance of empirical evidence attest to the fact that decision-making behavior is influenced by prior professional training, planned and accidental reinforcements of past decision-related behavior, on-the-job training, and other forms of cognitive or motivational "programming."

The Process Model emphasizes the effect of such programs and programming on organizational decisions. In its most rudimentary form, the model is captured by the following statement: *Organizational decisions are consequences of the programming and programs of the units involved (Proposition PRO).*

Another way of putting this thought, one that follows from the above statement, is that "organizational decisions at time T are predictable from a knowledge of the decisions at time T-1" (since programs and programming tend not to change quickly).

As with most statements of this nature, it is assumed that other relevant variables are held constant. However, the proposi-

tion does not preclude the influence of the variables central to three models of organizational decision making discussed earlier.

Let us be clear on two points at this juncture. One is that these four models are not the only representations of organizational decision making that appear in the organizational science literature. Other models have been developed that focus on particular types of situations, such as crises (see Snyder and Paige 1958, and Allison 1969) and capital allocation (see Cyert *et al.* 1979), or that demonstrate the implementation of a particular modeling technique (see Weber 1965, Smith 1968, Gerwin and Tuggle 1978). The four models discussed above are, however, the only conceptual models that have a broad empirical literature base; in sum, they do seem to capture the readily retrievable perspectives on organizational decision making.

The second point is that these four models are alternative ways of interpreting organizational decisions. In almost all important decision situations, *all four models have validity*. Part of what goes on may be interpreted with one model and part with another. The great majority of significant organizational decisions have their rational aspects, their political aspects, their predictable process aspects, and their chance or garbage can aspects. One model may be more explanatory in a particular instance than any one of the others; however, given many decisions, my guess is that less than half of the variance (in decision processes or outcomes) predicted by the sum of the models is predicted by any one of the models. A close reading of the descriptive literature on organizational decision making (see Allison 1969, Hah and Lindquist 1975, Mintzberg *et al.* 1976, Weiss 1980) certainly suggests that more than one model is necessary to describe what happens in most decision situations.

The main point of this discussion is that the designers of Decision Support Systems should be aware of the information requirements of each of these models, not just those of the Rational Model. If Decision Support Systems are more than simply efficient means of implementing OR/MS models, if they are really to be all that we claim that they are, then they must be designed in accordance with the manager's decision support system, a system that generally takes into account the informational requirements of all four models.

It seems reasonable at this point to examine the information requirements of each of the conceptual models.

INFORMATION REQUIREMENTS

The information requirements of the Rational Model have been enumerated as follows (Huber 1980c, Chapter 3):

Basic Information
- What are the alternatives?
- What are the future conditions that might be encountered?
- What are the criteria to be used in evaluating alternatives?

Elaborating Information
— What are the probabilities of the future conditions?
— What is the relative importance of the various criteria?

Performance Information
— What are the payoffs (or costs) associated with various outcomes?
— What are the constraints on the payoffs or costs?

For the purposes of this Task Force Meeting, we might want to add two others:
— What is the desired approach for dealing with uncertainty?
— What is the desired approach for dealing with conflicting criteria and constraints?

Some of the information requirements of the Political Model are detailed in Huber (1980c, Chapter 12).
— Who are the parties involved in the decision process?
— What is the potential influence of each of these parties?
— Which alternative does each of these parties favor?
— If a party favors our alternative, what are the ways that we can increase his or her influence?
— If a party is neutral, what are the ways that we cause him or her to favor our alternative?
— If a party favors a different alternative, how can we cause him or her to (a) favor our alternative, (b) lose interest in and withdraw from the decision process, or (c) lose influence in the decision process?

While at first glance these questions may seem unrelated to what we typically think of as a Decision Support System, a decision support system can answer them. Elements of a dss that might be captured in a DSS include the identities of members of any committees that are involved in the decision, the identities of the executives with authority for approving funding or enactment of the decision, the identities of the staff advisors of each of the committee members or executives, and the positions that each of these committee members, executives, and advisors are likely to take or have taken on similar decisions in the past. (As a former lobbyist at the Wisconsin State Legislature, I assure you that I had such information in my own dss.)

The information requirements of the Garbage Can Model might be a combination of those for the above two models, *if* the problems, the solutions, and the decision opportunities were under the control of the decision maker. However, the thrust of the Garbage Can Model is that the occurrence and pathways of problems, solutions, and decision opportunities are not predictable. Some examples of the information requirements of the Garbage Can Model are provided below:

Problems and Opportunities
— What are the problems and opportunities that may be forthcoming?

- How likely is it that these problems and opportunities will occur?
- When might they occur?

Solutions
- What are the potential solutions that may be forthcoming?
- How likely is it that these solutions will occur?
- When might they occur?

Choice Opportunities
- What are the potential choice opportunities that may be forthcoming?
- How likely is it that these opportunities will occur?
- When might they occur?

With a little imagination we can see that information about these matters could become part of a DSS data base, although this would require a more creative approach to forecasting than we typically find. It would be reasonable to expect that the raw data for such a data base might be obtained from a Delphi study rather than a multivariate statistical model.

The information requirements for the Process Model include a knowledge of the programs and programming of the parties and units involved in the decision process. Practical examples of such information are:
- the frequency distributions of the time necessary for the units involved in the overall decision process to produce the required information or recommendations;
- the expected level of quality or usefulness of this information or these recommendations, including the nature and expected magnitude of the biases that may be involved;
- the frequency distributions of the timeliness and effectiveness with which the decision-implementing units will carry out their assignments.

These three types of information are, without a doubt, extremely important components of the decision support system of any manager. To some degree it appears that they could be components of a Decision Support System as well. A noncreative example of this would be the inclusion in the DSS of the expected activity-completion times for a PERT or CPM chart. More creative examples would move us closer to designing the type of Decision Support System to which we all aspire.

SUMMARY

It seems clear from the above discussion that a manager's decision support system (dss) may at times reflect any of four models of organizational decision making. It also seems clear that Decision Support Systems will be more used and useful if they are compatible with the manager's existing decision support system, and if they are designed with due consideration to the knowledge that has been made available to us by organizational

scientists. While some of this knowledge has been highlighted in this paper, space limitations have permitted discussion of just a fraction of the knowledge available.

REFERENCES

Allison, G.T. (1969) Conceptual Models and the Cuban Missile Crisis. The American Political Science Review LXIII: 689-718.

Barnard, C.I. (1938) The Functions of the Executive. Cambridge, Massachusetts: Harvard University Press.

Bryce, J. (1888) The American Commonwealth. New York: AMS Press, Inc.

Cohen, M.D., J.G. March, and J.P. Olsen (1972) A Garbage Can Model of Organizational Choice. Administrative Science Quarterly 17(1): 1-25.

Cyert, R.M., and J.G. March (1963) A Behavioral Theory of the Firm. Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Cyert, R.M., M.H. DeGroot, and C.A. Holt (1979) Capital Allocation within a Firm. Journal of Behavioral Science 24(5): 287-297.

Downs, A. (1966) Inside Bureaucracy. Boston: Little, Brown & Company.

Gerwin, D., and F.D. Tuggle (1978) Modeling Organizational Decisions Using the Human Problem Solving Paradigm. The Academy of Management Review 3(4): 762-773.

Hah, C., and R.M. Lindquist (1975) The 1952 Steel Seizure Revisited: A Systematic Study in Presidential Decision Making. Administrative Science Quarterly 20(4): 587-605.

Hills, F.S., and T.A. Mahoney (1978) University Budgets and Organizational Decision Making. Administrative Science Quarterly 23(3): 464-465.

Howard, R.A., J.E. Matheson, and K.L. Miller (1976) Readings in Decision Analysis. Menlo Park, California: Stanford Research Institute, Decision Analysis Group.

Huber, G.P. (1980a) Organizational Information Processing Systems: Logistical Determinants of Performance and Behavior. Wisconsin Working Paper. University of Wisconsin, School of Business.

Huber, G.P. (1980b) Organizational Information Systems: Changes in the Form and Meaning of Messages. Wisconsin Working Paper. University of Wisconsin, School of Business.

Huber, G.P. (1980c) Managerial Decision Making. Glenview, Illinois: Scott, Foresman & Company.

Kaplan, M.F., and S. Schwartz (1977) Human Judgment and Decision Processes in Applied Settings. New York: Academic Press, Inc.

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

March, J.G., and H.A. Simon (1958) Organizations. New York: John Wiley & Sons, Inc.

March, J.G. (1962) The Business Firm as a Political Coalition, Journal of Politics 24(4): 662-678.

March, J.G., and J.P. Olsen (1976) Ambiguity and Choice in Organizations. Bergen, Norway: Universitesforlaget.

Mechanic, D. (1962) Sources of Power of Lower Participants in Complex Organizations. Administrative Science Quarterly 7: 350-364.

Mintzberg, H. (1975) The Manager's Job: Folklore and Fact. Harvard Business Review 53: 49-61.

Mintzberg, H., D. Raisinghani, and A. Theoret (1976) The Structure of 'Unstructured' Decision Process. Administrative Science Quarterly 21: 246-275.

Mobley, W.H., and B.M. Meglino (1977) A Behavioral Choice Model Analysis of the Budget Allocation Behavior of Academic Deans. Academy of Management Journal 20(4): 564-572.

Moch, M.K., and L.R. Pondy (1977) The Structure of Chaos: Organized Anarchy As a Response to Ambiguity. Administrative Science Quarterly 22(2): 351-362.

Perrow, C. (1979) Complex Organizations, A Critical Essay. 2nd edition. Glenview, Illinois: Scott, Foresman and Company.

Pettigrew, A.M. (1972) Information Control as a Power Resource. Sociology 6: 187-204.

Pettigrew, A.M. (1973) The Politics of Organizational Decision-Making. London: Tavistock.

Pfeffer, J., G.R. Salancik, and H. Lebleici (1976) The Effect of Uncertainty on the Use of Social Influence in Organizational Decision Making. Administrative Science Quarterly 21: 227-254.

Ritti, R.R., and F.H. Goldner (1969) Professional Pluralism in an Industrial Organization. Management Science 16(4): 233-246.

Schmidt, S.M., and T.A. Kochan (1972) Conflict: Toward Conceptual Clarity. Administrative Science Quarterly 17(3): 359-370.

Simon, H.A. (1947) Administrative Behavior. New York: Macmillan.

Slovic, P., B. Fischhoff, and S. Lichtenstein (1977) Behavioral Decision Theory. Annual Review of Psychology 28: 1-39.

Smith, R.D. (1968) Heuristic Simulation of Psychological Decision Processes. Journal of Applied Psychology 52(4).

Snyder, R.C., and G.D. Paige (1958) The U.S. Decision to Resist Aggression in Korea: The Application of an Analytical Scheme. Administrative Science Quarterly 3: 343-378.

Walton, R.E., J.M. Dutton, and T.P. Cafferty (1969) Organizational Context and Interdepartmental Conflict. Administrative Science Quarterly 14(4): 522-543.

Weber, C.E. (1965) Intraorganizational Decision Processes Influencing the EDP Staff Budget. Management Science 12(4).

Weiss, C.H. (1980) Knowledge Creep and Decision Accretion. Social Science Research and Decision Making. New York: Columbia University Press.

USING PERSONAL DATA BASES FOR DECISION SUPPORT*

Daniel Sagalowicz
SRI International

INTRODUCTION

Over the last few years a number of application systems that allow users to access data bases by posing questions in natural languages, such as English, have been constructed. When employed in restricted domains for which they have been specially designed, these systems have achieved reasonably high levels of performance. Such systems as LADDER (Hendrix *et al.* 1978), PLANES (Waltz 1975), ROBOT (Harris 1977), REL (Thompson and Thompson 1975), and EUPHID (Templeton 1979) require the encoding of knowledge about the domain of application in such constructs as data base schemata, lexicons, pragmatic grammars, and the like. The creation of these data structures typically requires considerable effort on the part of a computer professional who has had at least some special training in computational linguistics. Thus, because of the high cost involved in developing an interface to a particular data base, these systems are of limited utility.

Given capabilities for accessing data (remote and distributed data bases in particular), it is appropriate to accord more thoughtful consideration to the ways in which remote data are actually used. In many types of decision support tasks, decision makers require remote data to be augmented by local data bases (currently

---

*Preparation of this paper was supported in part by the Defense Advanced Research Projects Agency under Contract NOOO39-79-C-0118 with the Naval Electronic Systems Command. The views and conclusions contained in this document are those of the author and should not be interpreted as representative of the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency of the US Government.

recorded on paper or memorized) that store additional data or hypothesize situations other than those reflected in the actual shared data. We believe that the utility of natural-language systems for decision support applications will be greatly enhanced by allowing users to create local data bases easily and by automatically and appropriately directing queries to either the remote or local data base. The system, with help from the user, would reconcile conflicts in data values from the two sources.

In brief, our concept for near-term research to improve the use of data bases by decision makers is

-- to allow natural-language systems to be easily developed to access new, remote data bases and
-- to enhance the usability of the remote data by introducing auxiliary, local data bases.

In this paper we first present a prototype system, Transportable English Data access (TED), that allows a casual user to create and access remote as well as local data bases. We then explain the purpose and feasibility of enhancing the usability of shared data by introducing local, hypothesized data.

TED, A TRANSPORTABLE ENGLISH DATA ACCESS SYSTEM

The TED Prototype

TED is a natural-language processing system for accessing data bases, combined with an "automated interface expert" that interviews users to teach the language and logical structure associated with a particular data base and that automatically tailors the system for a specific application. TED allows users to create, develop, and edit their own new local data bases; to describe existing local data bases; or even to describe and subsequently access heterogeneous distributed data bases (as in Moore 1979).

Much of TED is based on components of LADDER (Hendrix *et al.* 1977). In particular, TED uses the LIFER parser and its associated support packages (Hendrix 1977), the SODA data access planner (Moore 1979), and the FAM file access manager (Morris and Sagalowicz 1977). In LADDER, all the data structures used by these components were manually generated by computer scientists. In TED, however, they are created by TED's automated interface expert. TED, like LADDER, uses a pragmatic grammar, but TED's pragmatic grammar does not make assumptions about the particular data base being accessed. It assumes only that interactions with the system will be about data access or update; other data structures, which are built by the automated interface expert, encode information about the particular data base.

The executive level of TED accepts three kinds of input: questions stated in English about the data in files that have previously been described to the system, questions posed in the SODA query language, and single-word commands that initiate dialogues with the automated interface expert.

Directions for Further Work on TED-like Systems

The TED system represents a first step toward truly portable natural-language interfaces with data base systems. However, as TED is only a prototype, much additional work will be required to provide adequate syntactic and conceptual coverage and to make adaptation of systems to new data bases easier.

A severe limitation of the current TED system is its restricted range of syntactic coverage. During coming months, however, we plan to remove these constraints by adapting Robinson's extensive DIAGRAM grammer (Robinson 1980) for use in a successor to TED.

The greatest challenge to extending TED-like systems is posed by the need to increase their conceptual coverage. As pointed out by Tennant (1979), users who are given natural-language access to a data base expect not only to retrieve information directly stored there, but also to compute "reasonable" derivative information. For example, if a data base has the location of two ships, users will expect the system to be able to provide the distance between them—an item of information not directly recorded in the data base but easily computed from the existing data. In general, any system that is to be widely accepted by users must not only provide access to data base information, but also enhance that primary information by providing procedures that calculate secondary attributes from the data actually stored. Data enhancement procedures are currently provided by LADDER and a few other hand-built systems. Work must now be done on devising the means for allowing system users to specify their own data base enhancement functions and to couple their functions with the natural-language component.

A second aspect of conceptual coverage is the ability to access information extrinsic to the data base per se, such as where the data are stored, how the fields are defined, and information about the status of the query system itself. User dialogues must also be developed for acquiring the information needed to answer such questions.

Systems such as LADDER are, then, of limited utility unless they can be created by individuals who lack significant formal training in computer science. While the development of user-specifiable systems with extensive conceptual and syntactic coverage is a research challenge, a polished version of the TED prototype, even with its limited coverage, would appear to have high potential as a useful tool for data access.

MANAGEMENT OF PERSONALLY EXTENDED DATA

Introduction

The systems now being developed, such as LADDER, TED, and others, enable a user to access a data base that is locally available or one that is remote and shared. If the local data base

is a copy of some subset of a remote data base, the user can use it in conjunction with the remote data base. This kind of coordination is performed automatically by our current systems to improve response time. When coordinating local and remote data, these systems assume that the data bases are consistent and that information can be retrieved from whatever data base is accessed most efficiently. Because some of our systems, such as TED, allow users to modify their local data base, answers to some queries may be inconsistent. No provision is made for handling inconsistencies in data that are redundantly stored in the two locations.

The assumption of consistency severely limits the combined use of local and shared data bases. Several key uses of large data bases require allowing the user to create a *personal* data base that is not consistent with the shared one. In such instances, the answers to queries should be consistent with the personal data base, even if most of the answers come from remote data.

It is therefore important to distinguish between the following concepts:

— local data: data that may be used locally without any attempt by the system to reconcile them with the shared data
— personal data: data used in conjunction with the shared data.

The current LADDER system can access *local* data, not *personal* data. Moreover, TED is able to generate local data interactively. The usefulness of such local data bases would be greatly increased if they could be integrated and reconciled with the shared data, i.e., if they could become personal data bases.

The following subsections describe typical examples of the use of personal data bases, as well as the technical difficulties that must be overcome to enable users to consistently use both a shared and a personal data base.

Example 1: Hypothetical Data Base

In many cases, high-level decision makers need access to the information in a data base to test various hypotheses. For example, a business executive may want to study the effects of introducing a new product on the revenues of a chain of stores. At present he could not easily set up a data base incorporating the new product. Because the modification is only hypothetical, it should not be "shared" among all users. The only hypothesis-testing technique currently available to him is to create a completely new copy of the data base, modify that copy, and then query it. This solution becomes infeasible as soon as the data base has attained a significant size.

A similar situation arises when a high-level decision maker wants to analyze the effect of continuing research on new drugs. Compounding the problem, he may want to test the impact of the new drugs on various hypothetical market situations, taking potential competitive reactions into account. As with the foregoing

example, the only hypothesis-testing technique currently available is to duplicate a large portion of the shared data base and to make the hypothetical changes in that copy, hoping that the result will be internally consistent.

Our approach is to facilitate the use of data bases for investigating hypothetical situations by allowing the user to set up a personal data base with only the changes incorporated into it. In the first example, the data base would contain only information about the new product. In the second example, the personal data base would encompass only the new drugs (as well as the new marketing environments affected by the changes).

The major difficulties involved in using personal data bases for testing hypothetical situations are reconciling personal and remote data and determining when personal data override remote data. To illustrate these difficulties, let us consider the second example and assume that the data base is organized as follows. In this data base, some of the drug characteristics are linked to the drug classes (many drugs belonging to the same class have the same characteristics). If the user is studying the introduction of a new drug A that belongs to an existing class C and wants to add new common class characteristics to be studied, the following straightforward technique will work. One record will be set up for class C with the new characteristics added, another for the new drug A in the personal data base. For every query that refers to drugs and drug classes, both the personal data base and the shared one will be accessed and the results will be reconciled.

On the other hand, if the user explicitly indicates that the new drug exhibits some significant differences for *some* of the common drug class characteristics, a more complex technique is required. In essence, some of the common characteristic fields of the C class record have to be "flagged" as exceptional cases. (Of course, the flag is set only in the personal data base.) For every new drug being studied, a record must be inserted into the personal data base. These new drug records would have all the attributes of both the old drug and the class records of the shared data base—in particular, the "common drug characteristic" attributes with the new values. For every query that refers to drugs and to their characteristics, the exception flag of the class record will have to be tested to ascertain whether exception drugs should be considered in the program's generation of an appropriate response.

Example 2: Decision Analysis

Decision analysis (Howard 1968) is a subfield of operations research that formalizes the analysis of decison problems. To perform such an analysis effectively, the decision maker must examine a set of hypothetical situations that further illustrate the previous category of examples. We distinguish decision analysis here because the set of difficulties to be resolved is slightly more complex than are those in the previous examples.

In a typical decision analysis session, decision makers attempt to evaluate the effects of a series of decisions, making some assumptions about the outcomes of each. The result is a *decision tree*, which is then analyzed by using operation research methodology. Our interest is in helping decision makers to evaluate the outcomes and the probabilities attached to each. This is, of course, similar to a series of hypothetical situations, as presented in the foregoing examples.

However, the interesting aspect of a decision analysis study is that it requires a multilayered organization of data. By this we mean that to manage a typical decision analysis situation we must be able to cope with several levels of hypothesis:

— The real world situation is at the bottom level.
— The next level represents the situation established by the decision maker when he is studying the first-level decision. In essence, this level models the world as if the first decision had been implemented.
— The third level represents the data for each outcome of that decision. In essence, this level models what is known about the world after each outcome has occurred; it is therefore clearly subdivided into as many subparts as there are outcomes of the decision.
— Finally, the next higher levels are obtained in turn by repeating the two previous steps for every decision level being studied.

Clearly, our techniques for handling personally extended data must allow for such multilayered situations.

Example 3: Summary Information

A decision maker may request the creation of a personal data base with summary information—for example, aggregating the financial condition of his organization. Depending on his needs, this summary information may be created and updated dynamically by the Data Base Management Software (DBMS), using the real shared data base, or it may be created by the user (or the DBMS) independently of the real data. The user could furnish the summary—unconcerned about the possibility of slight discrepancies in the real data—or request the use of summary information obtained previously but not kept up to date.

In such cases, the system must distinguish between summary and shared data. For example, if a user keeps the average salaries of departments in his personal data base and then requests all salary information for a department, the system ought to point out that the data may be inconsistent, i.e., that the numerical average may be different from the "summary average" stored in the personal data base.

Example 4: Replacement of Missing Data

A final example involves loss of part of the shared data base resulting from failures in the system, in communication, or in

both. In these cases, one may replace the missing data with other data (perhaps older, or even conjectured) believed to be nearly correct. This is similar to the hypothetical case of Example 1. However, should the missing data become available, the system—with or without help from the users—may have to reconcile the surrogate data with the real information.

The Approach

To study the construction of systems with *personal* data bases, we are taking a three-phase approach. Achievement of the first two phases has a high probability of success and should result in developing markedly improved capabilities. The third phase entails greater risk but also provides significant new benefits. The sections that follow briefly describe this three-stage approach.

Personal Subcopy of the Shared Data

In the first stage, we constrain the personal data base to be a copy of projections of the shared data base.* For example, let us suppose the user declares that two new drugs, A and B, have been put on the market—even though this fact could not be reflected in the shared data base. The personal data base in this case will contain a new relation, with fields for the drug name and characteristics. This relation will have a record for each drug in the shared data base and for the new additions. The system will automatically update its data structures so that this new relation is used for queries about drugs. The system we are developing is a "descendant" of LADDER. In this system the SODA schema, which provides a model of the data base structure, will be modified to indicate that those fields are to be found in the personal data base, not in the shared data base. The FAM directory, which includes a directory of files and DBMSs, will be updated to add the new relation.**

This approach will work only as long as the user's update does not force a change in the data base structure. The technique will fail if the personal data base requires a structure different from that of the shared one. In the drug and drug class example given previously, some of the drug characteristics are linked to drug classes in the shared data base but need to be linked to individual drugs in the personal data base. The shared data base still has two relations, the DRUG and DRUGCLASS relations, for example. The personal data base, on the other hand, will have only one relation, the DRUG relation, which includes the attributes of both the DRUG and the DRUGCLASS relations. As part of this

---

*In relational data base terminology, the projection of a relation, or file, is the relation obtained by keeping only a subset of the attributes, or fields, of the original relation.

**SODA and FAM, which are both parts of our current LADDER system, are described in Morris and Sagalowicz (1977) and Moore (1979).

first-stage effort, we are developing the conceptual structures necessary for recognizing such situations, as well as the procedures necessary for appropriate restructuring of the personal data base.

In sum, the personal data base in the first-stage approach possesses essentially the same structure as the shared data base. Whenever new fields need to be added to the personal data base, their relations correspond to the relations that have those fields in the shared data base. Only when relations need to be joined—as in the drug and drug class example—is that approach modified, as explained in the preceding paragraph.

Updating this kind of personal data base is handled in a straightforward manner. Every relation tuple updated by the user as part of his personal data base is flagged; those tuples are not updated in the shared data base. All other tuples in the personal data base are updated whenever their corresponding tuples are updated in the shared data base.

If one uses this particular approach, one should note that the system automatically accesses the personal data when required by a query; otherwise, it accesses the shared data. This automatic access accrues to us as a direct benefit of our current implementation of SODA and FAM.

This approach has two main drawbacks. First, the data are not consistent. To illustrate, let us again consider the drug example. There is no automatic attempt to verify whether it makes sense to remove existing drugs from the market. In the real world, the introduction of a new drug is likely to be accompanied by a complete reevaluation of the market. Many old drugs should be removed, or their use automatically restricted, in the new hypothetical situation. This type of "semantic consistency" will be studied as part of the third stage of our proposed approach.

The second drawback is that, if the relations to be copied have a large number of tuples, the approach results in a system that is unacceptably slow. This is the principal deficiency that we will try to eliminate in the second stage.

An Exception-Oriented Personal Data Base

In the second stage, we consider an alternative approach based on storing only exceptions in the personal data base. In the drug example, for instance, only those tuples that correspond to the new drugs would be entered into the personal data base. (In the drug and drug class example, we would store records only for those new drugs whose characteristics had been hypothetically assumed by the user and for those drug classes that have to be flagged as exceptional.) Then, when a request is issued (such as WHAT ARE THE PROJECTED COSTS FOR THE DRUGS WITH CHARACTERISTIC X ...?) the query will be sent to both the personal and shared data bases. In this case we will get two sets of answers for some of the drugs, the set that corresponds to the real world and the one that corresponds to the hypothetical world. The system must

then reconcile these data by discarding the real-world answers. This can be done quite simply: we eliminate from the answers all records originating in the shared data base when the same entities also appear in records originating in the personal data base.

Another technique we are studying in the context of this second stage is to issue a query that returns only appropriate responses. Continuing the drug example, we could first issue a query to the personal data base and then follow with one to the shared data base, asking explicitly that those drugs found in the personal data base not be included in the answer from the shared data base. In essence, the reconciliation would then have preceded querying the shared data base.

This second-stage approach is more complex than its first-stage predecessor. The data structures of SODA and FAM are less helpful because they do not provide for encoding information about exceptions. We therefore need to modify these subsystems so that they can use information specifying which relations may have exceptions. This information will be stored in the conceptual structures by the update routines when they perceive the existence of such exceptions.

As in the first-stage approach, updating the shared data base has to be reflected in the personal data, if appropriate. We handle this in the same manner as in the first-stage approach.

Although this alternative approach is more efficient, it shares one problem with the first-stage approach: neither guarantees the semantic consistency of the personal and shared data.

Consistency Handling

In the third stage of the proposed research effort, we are experimenting with the use of semantically based techniques to maintain consistency of the shared and personal data. Two categories of semantic checks can be distinguished; those that typify a particular kind of personal data base and those that are unique to each user.

Included in the first category are checks that should be performed when evaluating a hypothetical marketing situation. For example, introducing a major new drug in a hypothetical study should probably imply a corresponding realignment of its expected competitors. Checks related to summarizing information are another example of the first type of check. Thus, if a user keeps a summary of salaries, it should systematically be checked against answers he receives about the real salaries recorded in the shared data base. The summary may also be checked against the shared data base on a periodic basis, with the periodicity being specified by the user.

The second type of check, those unique to each user, is exemplified by the hypothetical drug and drug class situation,

in which the user modifies some common drug class characteristics. In such cases the user may want to make sure that the new drug characteristics replace the current ones, are added to them, or are combined with them. In this case, the system probably should simply remind the user of such problems and help him to set up the hypothetical situation in a consistent fashion.

The techniques we intend to use are totally open-ended. We are in essence following two methods.

In the first, we propose to include *personal data base functions* whose role is to maintain some type of consistency—for example, marketing consistency, in which the introduction of significant new products is always accompanied by a reorganization of the market of existing products, or summary data consistency. The problem with such special rules is that, by and large, they have to be rewritten for each situation. However, we believe that in many cases, such as the hypothetical product introduction and the data summarization, these rules would have wide application.

The second method is to base the processing on a uniform semantic representation whose expressive power will be sufficient to encompass information about all the layers of hypothetical data bases. Such an approach requires a powerful deduction mechanism and is therefore more distant in the future than the other techniques mentioned.

In general, a very large number of checks could be carried out to ensure the consistency of personal and shared data. Some of those checks would be unique to each user and could not be predicted by the system. Our research is concentrating on the mechanisms needed to handle consistency checks of a general nature. To perform checks that are user-specific, additional mechanisms, which enable the user to tell the system about new concepts and their interrelationships, must be provided. Clearly, research is needed to enable a casual user to introduce such knowledge easily.

SUMMARY

Our concept for data-base-related research to help high-level decision makers consists of

— expediting the development of natural-language systems capable of easily accessing and creating remote or local data bases and
— enhancing the usability of remote data through the introduction of auxiliary, local data bases.

We have briefly described a prototype system, TED, that allows a casual user to create and access both remote and local data bases. We have also briefly explained the purpose of enhancing the usability of shared data by introducing local, hypothesized data. In conclusion, we have presented some general

ideas on possible techniques for implementing such enhancements in practice.

REFERENCES

Harris, L.R. (1977) User Oriented Data Base Query with the ROBOT Natural Language Query System. Tokyo, Japan: Proc. Third International Conference on Very Large Data Bases.

Hendrix, G.G. (1977) The LIFER Manual: A Guide to Building Practical Natural Language Interfaces. Artificial Intelligence Center Technical Note 138. Menlo Park, California: Stanford Research Institute.

Hendrix, G.G., E.D. Sacerdoti, D. Sagalowicz, and J. Slocum (1978) Developing a Natural Language Interface to Complex Data. ACM Transactions on Database Systems 3(2): 105-147.

Howard, R.A. (1968) The Foundations of Decision Analysis. IEEE Transactions on System Sciences and Cybernetics 4(3).

Moore, R.C. (1979) Handling Complex Queries in a Distributed Data Base. Artificial Intelligence Center Technical Note 170. Menlo Park, California: SRI International.

Morris, P., and D. Sagalowicz (1977) Managing Network Access to a Distributed Data Base. Berkeley, California: Proc. Second Berkeley Workshop on Distributed Data Management and Computer Networks. Pages 58-67.

Robinson, J.J. (1980) DIAGRAM: A Grammar for Dialogues. Artificial Intelligence Center Technical Note 205. Menlo Park, California: SRI International.

Templeton, M. (1979) EUFID: A Friendly and Flexible Frontend for Data Management Systems. San Diego, California: Conference of the Association for Computational Linguistics.

Tennant, H. (1979) Experience with the Evaluation of Natural Language Question Answerers. Tokyo, Japan: Proc. Sixth International Joint Conference on Artificial Intelligence. Pages 874-876.

Thompson, F.B., and B.H. Thompson (1975) Practical Natural Language Processing: The REL System as Prototype. Pages 109-68, Advances in Computers 13, edited by M. Rubinoff and M.C. Yovits. New York: Academic Press.

Waltz, D. (1975) Natural Language Access to a Large Data Base: An Engineering Approach. Tbilisi, USSR: Proc. Fourth International Joint Conference on Artificial Intelligence. Pages 868-872.

DATABASE TECHNOLOGY IN DECISION SUPPORT SYSTEMS: AN OVERVIEW

Frank A. Manola
Computer Corporation of America

INTRODUCTION

This paper describes how database technology can contribute to Decision Support Systems (DSS), both in terms of present knowledge and future research and development. Since the author's acquaintance with DSS is rather limited (command and control systems being the closest type of system with which the author has been involved), no attempt is made here to state which facilities a DSS should provide. However, it was necessary to assume certain facilities (based on limited reading in the DSS literature) in order to determine which facilities may be provided to DSS by database technology. These capabilities are either already addressed by database technology or are in a research and development phase. Readers more familiar with the requirements of DSS can judge for themselves the extent to which the DSS requirements identified in this paper are in fact DSS requirements, and thus judge the applicability of database technologies that may be related to those requirements.

In the context of relating database technology to DSS, database technology can be divided into two classes:

— Aspects of database technology of direct interest to DSS.
— Aspects of database technology of only indirect interest to DSS. Security, recovery facilities, etc. are important to providing effective and reliable database facilities, but do not appear to be directly relevant to DSS.

There are two types of relationships between database technology and DSS:

— Some areas involve providing better *database* support to DSS. That is, how databases can be better used by DSS, or how

database systems can better support the specific requirements of DSS—DSS being considered one of many database applications. This involves a very direct relationship between DSS and database technologies.

— Other areas involve cases in which database technology can help provide better DSS capabilities directly, whether or not a database *per se* is involved, i.e., spinoffs from database technology may assist in some DSS functions. Examples are cases where both types of systems face (and must solve) similar problems.

DSS REQUIREMENTS

Blanning (1979) suggests that a DSS may perform one or more of the following six functions:

Selection of data from a database. According to Blanning this function is not as widely used, mainly because DSS requirements tend to involve data analysis rather than data retrieval. However, the data must be retrieved before it can be analyzed, and Blanning notes that database management systems and query languages can facilitate this function. (This function, of course, suggests that *any* areas of database technology that contribute to the improvement of database system power or efficiency contribute indirectly to DSS.)

Aggregation of data into totals, averages, frequency distributions, etc., which can help identify problem areas for further investigation.

Estimation of the parameters in a probability distribution; this can be accomplished by performing statistical analyses of data with the aid of statistical packages or interactive data analysis packages.

Simulation to calculate the anticipated consequences of proposed decisions and/or of possible changes in a corporate environment. Numerous software packages already exist for assisting in the simulation process.

Equalization to calculate decision strategies whose consequences will meet certain consistency conditions, using, for example, sets of simultaneous linear equations or economic models.

Optimization to determine decisions that will maximize or minimize a single measure of performance or cost without violating constraints on other measures. According to Blanning capabilities similar to those used in simulation, plus interactive matrix generation and report writing languages, can be used as computer support for this function.

Based on these DSS functions, a number of general software- and data-oriented requirements can be identified. These include:

(1) The ability to construct accurate models of an enterprise, including entities of interest within the enterprise, their interrelationships with one another, and consistency constraints that govern their interactions. While these models may be highly mathematical in many cases, and expressed as equations, they may also involve explicit declaration of types of entities and their attributes (as in a simulation).

(2) The ability to retrieve data from a database, using a relatively high-level and easy-to-use query facility.

(3) The ability to easily and flexibly apply various types of special-purpose software packages to data obtained either from a database, or as output from some other software package. This requires not only flexible software interface capabilities, but also the ability to save—in a relatively straightforward way—the output from one package so that it can be used as the input to another package. The subsystems needed to perform data analysis and other functions, as well as the data required for input to these subsystems, may exist in different parts of an organization, or even outside the organization altogether. The subsystems may have been developed independently. The data may have been collected for different purposes, and may be inconsistent, in different formats, etc. (see Nash 1977).

(4) The ability to specify the format of the output of an analysis or retrieval so that it is appropriate for user purposes. The mention of query and report writing capabilities in Blanning (1979) illustrates this requirement.

In addition, much of the literature surveyed for this paper indicates general agreement on a number of other requirements. These requirements include:

(5) The ability for users to easily determine which data and software tools are available in their environment for use in solving DSS-related problems. This includes not only the names of data and programs, but also the meaning of the data, the function of the programs, program input and output requirements, etc. (see Carlson 1979).

(6) The ability to produce output in the form of reports, charts, diagrams, pictures, video displays, etc.

(7) The ability for software to provide human-engineered computer interfaces for decision makers.

This includes the use of graphics facilities as input, interactive tutorial assistance, etc. (Buneman and Clemons 1979, Carlson 1979, Nash 1977).

(8) The ability of systems to provide "triggers" or "alerting" capabilities; i.e., the ability of the system to monitor specified conditions and notify the user when they occur.

(9) The ability to provide users with access to other large databases, libraries, or software tools that may exist, even though they may not be available at the user's location.

(10) The ability to provide effective communications facilities to decision makers when more than one person is involved in a decision.

Finally, the following is a requirement that is often not explicitly mentioned, but that is added here for completeness:

(11) The ability of software tools to perform efficiently. It is a familiar idea that because of their speed computers have enabled problems to be tackled and procedures to be used, which were previously impossible. Improved computer system performance may continue this process, permitting the evaluation of more data, the consideration of more alternatives, and more effective human interface in DSS processing.

If the above identification of DSS functions and the derivation of DSS computer-oriented requirements are correct, then it would be of great use to enhance the ability of DSS to meet these 11 requirements. The next section describes how database technology might contribute to satisfying these requirements.

DATABASE TECHNOLOGY

The requirements identified above for DSS are to a great extent similar to requirements imposed increasingly on database systems (DBMS). Database technology does not, perhaps, attack these requirements in the same directions and with the same point of view as DSS technology *per se* would, but it is likely that the approach taken by database technology toward meeting these requirements may contribute to the solution of these problems in the DSS area. In the following sections, a number of specific aspects of database technology are briefly described that directly contribute to the solution of the identified requirements, namely

— Data models and database system architecture
— Data translation and mapping
— Database access languages
— "Active" DBMSs
— Distributed database systems
— Database hardware

The relationship of each of these areas to the solution of the specific DSS requirements identified earlier will be discussed below. The following table summarizes the connection between the aspects of database technology that will be described and the numbered DSS requirements discussed earlier.

Area of Database Technology	Related DSS Requirement
Data models and database system architecture	(1), (5), (11)
Data translation and mapping	(3)
Database access languages	(2), (3), (4), (5), (6),(7)
"Active" DBMSs	(3), (8)
Distributed database systems	(3), (9), (10), (11)
Database hardware	(11)

The discussion of database technology in the following sections (and many of the references) stems in part from Morgan *et al.* (1980) and Yao *et al.* (1978). These are useful surveys of database technology in general and include areas not specifically covered in this paper.

Data Models and Database System Architecture

The user of a database interprets the real world outside the database in terms of a set of 'real world' objects (entities and relationships between entities) and actions involving these objects. The user interacts with the database in order to derive information that he needs or to store information about the 'real world' for his own use or for others'. However, a DBMS does not operate in terms of the user's 'real world' objects and operations, but rather in terms of data objects (data elements and relationships between them) and operations on the data objects; therefore, in order to use the database the user must perform a transformation or *mapping* from his perceived 'real world' objects and actions to those of the database. The *data model* determines the type of data objects and operations that the user sees as he interacts with the database. Using the data objects and operations defined within the model—and under the restrictions imposed by the rules of the model—the user attempts to model the 'real world', and actions in it. The relevant 'real world' entities and relationships, and all their attributes, must first be described in terms of the legal data objects and data attributes of the model. This is done using a *data description language (DDL)* based on the data model. The collection of DDL declarations that define the data contents of the database is referred to as the database *schema*. In order to model some action or sequence of actions on 'real world' objects, the user must map these actions to a sequence of operations allowed by the data model. These operations must then be submitted to the DBMS using a *data manipulation language (DML)* based on the model—for example, in a program or as a sequence of queries. The result of this sequence of operations presumably corresponds to some 'real world' result.

Almost all commercially-available DBMSs are based on some version of three main data models: the relational (Codd 1970), hierarchical (Sibley 1976), and network (CODASYL 1971, CODASYL 1978b) models [see also Date (1977) for a discussion of data models]. The basic concept of a *record* is common to these models. The models also impose various constraints on the internal structure of records and the way they may be related to one another to represent real-world relationships. (For example, a department record and an employee record may be related to represent the employee's assignment to that department.) In order to use a particular DBMS, the user must translate between his own view and that of the particular model used in the DBMS.

One of the original reasons for the development of data models was to spare users concern over the details of dealing with various kinds of hardware storage devices. Accordingly, the data model with which the user interacts is generally not one of *physical* (i.e., oriented toward representation on devices) data objects (such as tape reels, disk tracks, etc.) and operations, but rather one of *logical* (abstract) objects and operations. In Figure 1 the "schema" represents the objects of the logical data model with which the user directly deals. The user performs the "mapping" between the user and logical data. Ultimately, the logical data with which the user interacts must be mapped onto physical data stored on devices; in the same way, the user's operations on logical data must be mapped onto operations on the physical data.

Just as a mapping is performed by the user between his 'real world' objects and operations and the logical data objects and operations, the DBMS must perform a mapping between the logical and physical data objects and operations.

The separation of the user from the physical structure of the database provides the potential for *physical data independence*, i.e., changing the physical structure of the database without affecting the logical structure seen by the user. (The logical physical mapping must also be changed, but the user is not involved in this process.) A related concept, called *logical data independence*, will be discussed later.

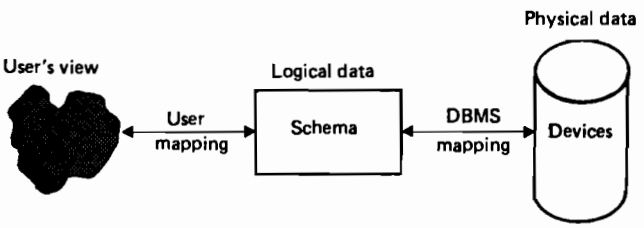


FIGURE 1

The complexity of the mapping that a user must perform to connect his 'real world' view to logical data naturally depends on the degree to which the objects and operations of the logical data model match 'real world' objects and actions. If there is little correspondence, then the mapping will be extensive. This may imply great difficulty in representing the 'real world' in the model represented by the database (or even the inability to represent certain key aspects of the real situation in the database). The extensive mapping may necessitate many operations (a large program or sequence of query statements) to perform simple 'real world' actions, or require much mental effort on the part of the user to perform database operations and understand results received from the database (which in turn may lead to errors on the user's part). In a similar manner, the difficulty of mapping between logical data and physical data depends on the degree to which the logical data structures and operations match physical data structures and operations. Little correspondence may mean inefficiencies of various kinds—for example, large storage requirements or processing overheads.

Any logical data model represents, to some extent, a compromise between the goals of ease of use and naturalness of expression for users, and ease and efficiency of mapping to physical representations. The three major data models cited above tend to be close to the physical end of the spectrum, since they are essentially based on abstractions of physical storage objects and operations. This implies that in dealing with these models the user's mapping can sometimes become complex. [The problems associated with these models have been explored by Kent (1979).] However, two developments have combined to simplify the user's problem.

The first development, which is already incorporated in a number of commercially available DBMS products, is the use of logical subsets of the schema, tailored to the specific data requirements of different users. These logical subsets, called *subschemas* (CODASYL 1971, CODASYL 1978a), are DDL descriptions of the particular portion of the database in which a given user is interested. When the user accesses the database through a subschema rather than a schema, he may deal with a less complex data structure. Also, the subschema protects the rest of the database from possible mistakes by the user. In addition, separation of the user from the logical structure of the database as a whole (described in the schema) has other advantages. It provides the potential for changing parts of the database logical structure that are not referenced in the user's subschema, without affecting the logical structure seen by the user in his subschema. This capability is known as *logical data independence*. Naturally, changes in parts of the database structure that the user has to see must be reflected in his subschema, and these changes may change his operations on the database. Figure 2 illustrates the provision of subschemas by the database system. Here the user's mapping is simplified by removal of irrelevant logical data from the subschema.

The second development has yet to be reflected in commercial DBMS, but has been incorporated into some prototype systems. It

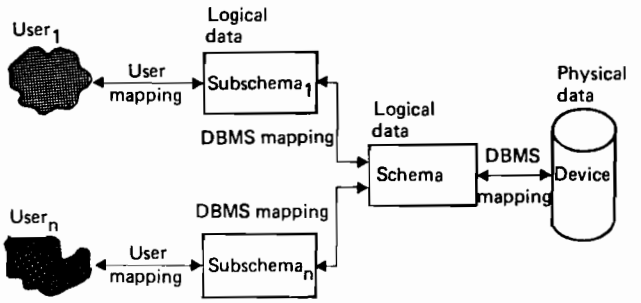


FIGURE 2

is the use of increasingly abstract data models that contain constructs and operations more closely representing the properties and behavior of 'real world' objects. These models are easier for users to understand; they also represent more accurately key aspects of the 'real world' situation modeled by the data. Such models are often referred to as *semantic* or *conceptual* models, and the schema based on such a model is called a *conceptual schema*. Examples of conceptual models may be found in Chen (1976), Codd (1979), Hammer and McLeod (1978), Nijssen (1976), Nijssen (1977), Senko (1976a), Senko (1976b), and Smith and Smith (1977). The same trend is occurring in programming languages, where more abstract data objects are referred to as *abstract data types*.

The development of such schemas has led to the concept of a *DBMS architecture* or *framework*; for the purposes of this paper, this framework may be considered a description of the types of schemas that may be defined under the system and the way in which these schemas are related by mappings. By defining the allowed types of schema, the framework describes the various types of interfaces to the DBMS, which the system supports. Figure 2 describes a two-schema framework, for two types of schemas (logical schema and logical subschema) are defined in it. Perhaps the most-discussed DBMS framework in recent years has been a framework proposed by a study group of the American National Standards Institute (ANSI 1975, Tsichritzis and Klug 1977). This three-schema framework incorporates a central *conceptual schema*, a generalized type of subschema called an *external schema*, and an *internal schema* for specifying explicitly the database storage structure to be used in supporting the conceptual model. [All three of these terms were introduced in ANSI (1975).] Figure 3 illustrates this three-schema framework. A similar framework has been adopted in the recent CODASYL database specifications (CODASYL 1978a, CODASYL 1978b), which are currently undergoing standardization by ANSI technical committees.

In the three-schema framework, the external schemas define the logical subset of the database to be seen by a given user; the conceptual schema defines a conceptual model of the enterprise

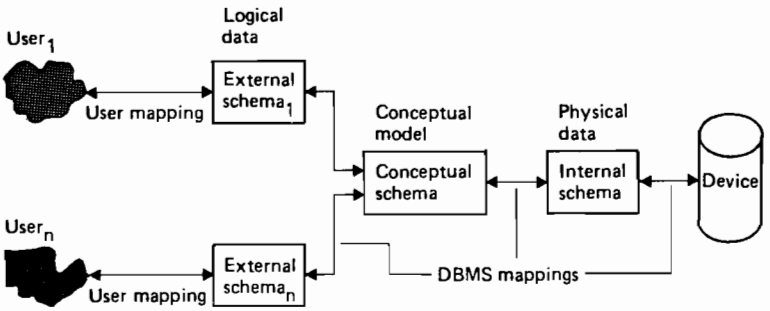


FIGURE 3

whose operations are represented by the database; and the internal schema defines the physical data structure that supports the conceptual model. Correspondence between objects and operations at these various levels of description is established by the mappings. The mappings bind descriptors in one schema to those in another. It must be possible to map external and internal schemas to the conceptual schema, in order to guarantee consistency with the conceptual model.

More recently, frameworks with additional types of schemas have been proposed, in connection with the requirements of distributed database systems. However, these schemas are not strictly relevant to the present discussion.

The explicit data declarations represented by the various types of schemas in database systems constitute documentation about the database, which can be useful to users. It is expedient to have a central repository where these declarations and other "data about data" can be kept. Such a central repository is called a *data dictionary/directory (DD/D)*. This component is at the heart of the ANSI framework, acting as a repository for both the schema definitions and the definitions of the mappings between them. Recent prototype systems (Date 1977) allow such "meta-data" to be queried in the same way as the database itself. This facility makes the user interface far more flexible, and also makes it self-documenting to some extent, since the user can ask about the availability of data, and other information about the data that may be stored in the DD/D (such as where the data comes from). Research on more general types of DD/D systems is being actively pursued (British Computer Society Data Dictionary Systems Working Party).

The above discussion suggests that both a DBMS and a DSS require an accurate model of the enterprise they are intended to model. The more information that such a model can convey to a computerized tool such as a DBMS or DSS, the better job the tool will be able to do in providing accurate information in support of user requirements.

Research into advanced data models can directly contribute to the success of decision support systems. These data models

permit DBMS, which act as components of such systems, to provide more accurate data, and allow DBMS users to interact more easily with the database. Such research can also indirectly affect decision support systems, by improving modeling capabilities used in components of a DSS other than the DBMS. For example, data modeling capabilities provided in such simulation languages as SIMULA (Birtwistle *et al.* 1976), to simplify the process of defining the enterprise to be simulated, had an early impact on the development of abstract data types in programming languages. In many respects current research on data models parallels the work on abstract data types. As a result, it seems reasonable to expect that research on data models will have an effect on future DSS components that involve modeling.

The use of existing data models in DBMS frameworks (such as those described above) has already contributed to the efficiency and effectiveness of DBMS, including their use in DSS. The ability to specify the database structure explicitly in a schema is crucial to the ability of an organization to centrally manage its data resources; it is also crucial to the ability of the DBMS to control access to this data. The ability to specify subschemas helps to simplify user access to databases, and also to provide security and integrity for other users. The ability to specify physical database structures independently of the schema will be an important factor in improving database performance without adding to user complexity, for complex physical structures may then be specified independently of the logical structures seen by users.

The need for DSS users to be able to determine the availability of data and software tools in their environment parallels the needs of database users that led to the development of DD/D systems. As typically used, DD/D systems contain descriptions of database data, of data in files within an organization, and of program resources. In addition, they may also contain descriptive information about the data in those files (and databases), such as how it is collected and its degree of accuracy. As DD/D systems become more widely used and contain more complete information about data and program resources within an organization, they will become valuable DSS components. This is especially true if the DD/D can be queried by the user with a high-level query language. This capability allows those responsible for assembling DSS components to quickly determine the resources that are already available within their organization. Future development along these lines might involve the use of computer networks to allow organizations to "borrow" each other's software and data (under organizational control, of course); these components could interact over the network. In this scenario, someone wishing to construct a DSS might then query not only his own organization's DD/D, but also those of other organizations on the network, for useful software and data.

Data Translation and Mapping

The different types of schemas represented in the database frameworks described previously have made the problem of mapping

between the various schemas increasingly complicated. As long as the subschema was a logical subset of the schema, the mapping was relatively straightforward. However, with the introduction of more conceptual models, and the use of specialized internal and external schemas, the mapping becomes rather difficult, and specialized languages (or parts of DDLs) have been constructed in which to specify these mappings (Bonczek and Whinston 1977, CODASYL 1978b, Shu *et al.* 1977). This problem of mapping between schemas is similar to the problem of translating data from one form to another, which has also received considerable attention recently. Programs to perform data translation have been used throughout the history of computing to help move data between different environments. Incompatibilities in hardware and software conventions have always meant that a change in either could make data produced under one convention unusable under another. The earliest translation programs tended to be one-shot programs for translating a single file to a new form. However, the increasing cost and frequency of system conversions and the difficulties introduced by the advent of DBMSs forced the rethinking of this approach. At that time, the need to convert data from existing file structures to formats suitable for DBMSs stimulated research into generalized data translators (Fry *et al.* 1972, Smith 1975, and Taylor 1971). This research has produced a number of experimental prototypes (Fry *et al.* 1972, Ramirez 1973, Shu *et al.* 1977). Commercial systems have also been built and studied (Bakkom and Behymer 1975, Sperry Rand Corp. 1974, Winters and Dickey 1976).

A generalized data translator resembles to some extent the DBMS frameworks described above (see Figure 4). The user supplies a specification of the source and target data structures in a given DDL and a specification of the desired mapping between them. When data conforming to the source specification is used as input, it is converted into output conforming to the target specification. (The DDLs used in data translation tend to be more complicated than those used in logical data description, for the exact physical formats of the data at both ends of the translation must be specified in data translation.) To date, translators have not demonstrated capabilities for dealing with all the structures that are definable using current logical data models. Still, a substantial subset of such structures has been considered and further progress is certain.

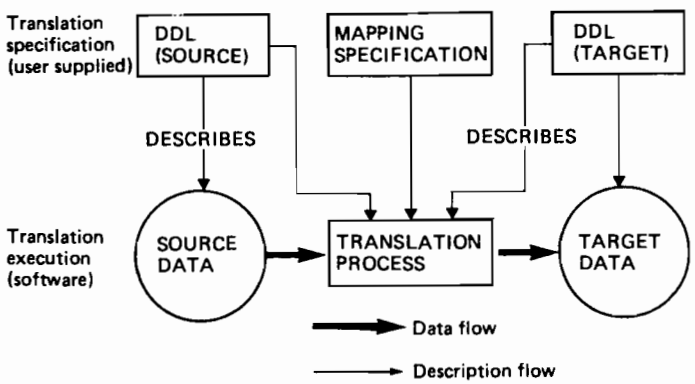


FIGURE 4

As is suggested by the juxtaposition of topics, investigations of data translation and data mapping complement one another. The mapping specification problem is similar in both environments, and solutions in one area constitute solutions in the other. (In fact, the difference is largely one of implementation; mappings must be done on demand, while translation is usually done in bulk.) Data translation technology is also needed when databases are restructured due to changing application requirements. Furthermore, data translation takes on a new importance within the context of a distributed database system. Even in a network in which all the interconnected databases use the same data model, different sites may require different logical structures; this in turn necessitates translation (or mapping) when such sites communicate. The problem becomes more complex when the network is heterogeneous, and when different data models are used at different sites. Research has begun on extending data translation technology to this problem, but it is still at a very early stage (see Schneider and Desantels 1975).

Database program conversion is a related spinoff of data conversion technology. For example, in moving from one database environment to another, changes may be made in the logical structure, data model, or physical structure. Such changes can require that significant changes be made to an accessing program, before the program can run with equivalent results in a new environment. In organizations with large databases and thousands of application programs, the cost of making these changes manually can be so large that system conversion becomes infeasible. This problem is only beginning to receive attention (Housel 1976, Shneiderman 1978b, and Su and Liu 1977).

Developments in data translation and mapping can considerably simplify the problem of integrating independently developed software and independently collected data into useful decision support systems. For example, a collection of data analysis tools could be applied to a given set of data in series: the data could be converted to the form required for input to the first tool; the output of that tool could be converted to the form required for input to the next tool, and so on (assuming, of course, that the data is suitable for the application of such tools). The mapping capabilities of the DBMS could be used to allow such software to access database data either directly, via special external schemas, or indirectly, via intermediate files produced by the DBMS. If the intermediate files are not themselves in the required form for input to the software, a data translation program could be used to provide the proper form (see the following section). Alternatively, program translation technology, if successfully developed, could be used to adapt the program to the data, rather than vice versa. In some circumstances, this may be the most efficient or economical approach. The flexibility provided by these developments would assist in making programs and data more adaptive to user requirements, rather than users having to adapt to the software.

Database Access Languages

As described earlier, the user operates on a database using a data manipulation language, or more generically, a *database access language*. (Strictly speaking, a data manipulation language (DML) is a specific type of database access language.) The operands of database access languages are the data objects defined in the schema or subschema being used by the user, as defined in turn in the DDL. The general types of operation provided by the database access language include *retrieval* of data objects, *storage* of new data objects, and *modification* or *deletion* of existing objects and relationships.

Three classes of users have been distinguished in the development of current database access languages: applications programmers, technical users, and parametric users. The applications programmer is a professional programmer who is responsible for producing and maintaining the programs that operate on the database. Database access languages oriented toward the programmer's needs are called *data manipulation languages*. These are generally embedded within programming languages such as COBOL or FORTRAN (CODASYL 1978a). The technical user is one whose primary interaction with the database is retrieval. The technical user's information needs may be unpredictable in the sense that he does not frequently repeat the same retrieval request. The technical user is trained in the use of special database access languages called *query languages* (Chamberlin *et al.* 1976, Codd 1971a, Codd 1971b, Shipman 1979, Sperry Rand Corp. 1977, and Stonebraker *et al.* 1976). The parametric user has fixed and regular interactions with the database. Sales clerks and airline reservation agents are typical parametric users. The parametric user uses special interfaces that provide a fixed set of simple options that require limited learning; such interfaces may even operate by prompting. They are typically designed and implemented by applications programmers using data manipulation languages.

The emphasis in database access languages has been shifting from data manipulation languages to query languages, and from applications programmers to the technical users and so-called "casual users." A casual user is a user "whose interactions with the system are irregular in time and not motivated by job or social role" (Codd 1970). Currently, when new languages are developed for database applications programming, they tend to contain high-level operators similar to those found in query languages (Shopiro 1979). This shift in emphasis requires query languages that are more powerful (for the technical user) and easier to use (for the casual user). A considerable amount of research is directed toward the development of query languages and other database interfaces with these attributes. Approaches investigated to provide easier-to-use database interfaces include the following:

(1) Conventional query languages are being analyzed to determine operators or syntactic approaches that cause users difficulties (Reisner 1977, Shneiderman 1978b, and Thomas 1977).

(2) The use of graphics to simplify interfaces is being investigated. Interfaces have been developed in which a "query" is expressed as a graphical shape (McDonald 1975), by a series of light-pen touches to a graphical diagram of the database schema (Senko 1976a, 1976b), or by user specification of samples of the output required in a tabular format (Zloof 1975). Still another approach is taken by Herot (1979); here a spatial data management system provides graphical symbols for the data itself, rather than for query language primitives or schema objects. For example, ships are represented by outlines of ships, the Pacific Ocean by a patch of blue on a map of the earth's surface, and so forth. The user "queries" the system by moving over the graphical symbols until the appropriate object is found. The user can also "zoom in" on a symbol, to obtain more detailed data about the entity in question, or "zoom away" from a symbol, to obtain less detail about a particular entity, but information about "nearby" entities.

(3) The use of an English-like dialogue capability is also under study. In this case the user poses his request in English, and the system attempts to interpret it, using artificial intelligence techniques for natural language processing, and dialogue with the user to resolve any ambiguity. Codd (1978), Harris (1977), and Hendrix *et al*. (1978) give examples of this approach.

(4) The use of special terminal facilities, such as light pens, function keys, joysticks, or trackballs, is being explored to simplify specification of requests. These special facilities are often used in combination with one of the approaches already mentioned. For example, a graphics-oriented approach frequently requires special terminal equipment.

Capabilities investigated in adding power to query languages may be summarized as follows:

(1) Enhanced capabilities for the user to define the format of the query output are being developed. Such output may need to be in the form of a report with a complex structure, or a visual display. Another form of output may be actual stored data, either in a form which can be used as input to subsequent queries, or saved in an external file that can then be used as input to nondatabase programs.

(2) Attempts are being made to provide text-searching capabilities in the query language. This is an increasingly important function as database systems are increasingly used in roles previously reserved for specialized information retrieval systems. Large amounts of text data are being stored on database systems. However, without built-in text processing capabilities, storing and searching for such data can become extremely inefficient, as well as burdensome to system users. DBMSs are beginning to provide limited text searching capabilities (Computer Sciences Corp. 1977 and Software House 1976). Such systems may be extended in the future, along the lines of information retrieval systems, to include thesaurus capabilities that permit retrieval on "related terms" and other specialized facilities.

(3) Finally, browsing capabilities are being developed as an easy-to-use database interface. A browsing capability allows a user to rapidly scan data in a database in order to look for interesting or relevant material. It is more important to casual users than to technical users. Browsing does not simply involve rapid response from the DBMS, although this is certainly important. Browsing also requires abstraction facilities (because too much detail can overwhelm the user), and "rapid transit" facilities to enhance free-association on the user's part. Herot (1979) provides an example of a system with a framework for supporting browsing.

Developments in database access languages address a number of the DSS requirements identified earlier in this paper. Specifically, these languages address requirements for powerful, high-level, easy-to-use data access interfaces, which can produce output in a form tailored to specific user requirements—whether for hard-copy reports, graphics displays, or in the form of stored data for input to another program, such as a data analysis or simulation routine. In addition, knowledge gained from investigations into the human aspects of database interfaces should provide useful input for the design of interfaces for other types of computer systems. The relationship of this work on database interfaces to the DSS environment is particularly close; the "casual users" for whom many of these advanced interfaces are being developed are in many cases the decision makers to whom decision support systems are also addressed. (And developments in high-level database interfaces have often been prompted by the demands of military command and control systems—many of which contain components of DSSs, if they are not DSSs themselves.)

"Active" Data Base Management Systems

The idea behind an "active" DBMS is that the DBMS is capable of taking more or less independent action when user-specified conditions arise, rather than passively executing storage and retrieval requests. The user-specified conditions may be simple, such as "when an update to the employee record is performed" or complex, such as "when the stock-on-hand for any part falls below the reorder level." Correspondingly, the actions to be taken may be simple, such as "print a message at my terminal" or complicated, involving storage, retrieval, or update of database data by the DBMS itself or the execution of some independently written program available to the DBMS in a procedure library.

Simple facilities along these lines are provided in the CODASYL database specifications (CODASYL 1978b). Procedures, called *database procedures*, may be named in the database schema and executed when specified database operations are performed on specified data objects. A similar capability, called a *trigger*, is described in Astrahan *et al.* (1976). Buneman *et al.* (1977) present a more complex facility, called an *alerter*. In an alerter, the condition to be monitored can be rather complex, but the action is usually simple (i.e., notify a user). This differs from the "active" DBMS discussed above, where the condition is usually

simple, but the action is the execution of a (potentially complex) program.

Such active DBMS facilities can be used to notify users of a database system when certain critical situations occur. In certain cases, they can also be used to take corrective action if the necessary action can be anticipated. In addition, they can be used to prevent certain actions. For example, they can be used to make sure that attempted updates to the database are "legal" (satisfy specified database integrity constraints) and refuse to allow illegal actions (Stonebraker 1974). Such facilities can also be used in the generation of *derived data*, i.e., data that may not be stored in the database, but that can be computed from data that is stored in the database. Derived data may be presented to a user in his external schema as if it were actually stored, when in fact it is computed on request. (Of course, a program must be invoked when retrieval of such data is attempted so that the data values can be computed.) An example of such derived data might be the current estimated location of a ship, computed on request using the time of request and the last known position, course, and speed of the ship.

Active DBMS capabilities can be useful in DSSs in a number of ways. First, these capabilities can be used for alerting a user when some condition occurs. Another example would be the situation in which a decision maker does not want to see raw data in a database, but rather the results of an analysis of the data performed by one or more data analysis programs. Using derived data capabilities, the programs could be added to the DBMS program library and then invoked to produce analyzed data on request. Programs for producing data statistics can be handled in the same way. (In fact, such programs are already built into many database systems.) If, for some reason, it were not appropriate to incorporate such a DBMS directly into a DSS, similar capabilities could be constructed for nondatabase software.

Distributed Database Systems

A distributed database system (DDBMS) allows data to be stored at multiple locations in a computer network, yet to be accessed as a single unified database. A DDBMS offers the following potential advantages over a centralized DBMS:

— It can be made more reliable (provided suitable data replication is used), since multiple computers at multiple locations are involved.
— Data can be stored at locations where it is frequently used, resulting in faster access and lower communications costs.
— System capacity can be easily adjusted to meet changing needs by adding additional network nodes.
— Performance can often be improved by allowing multiple computers in the network to operate in parallel on different parts of a single request.

The initial users of DDBMSs are likely to be large, geographically dispersed business and government organizations. Such

systems will foster coordination within organizations by providing managers with timely access to data dispersed throughout the organization. Moreover, the development of the underlying technology of these systems provides a basis for the interconnection of computers that support many different types of data; the computers may then be used together in unanticipated ways in various decision-making activities, even though the computers may not necessarily belong to the same organization.

The potential advantages of DDBMSs, together with increasing development of the required technology, have generated a substantial amount of interest (see Rothnie and Goodman 1977). Considerable development of the communications technology underlying the DDBMS has taken place (for example, the Arpanet computer network in the USA). Champine (1977), Foster (1976), Pliner *et al.* (1977), and Wiseman (1977) report on a number of DDBMS implementations that are quite large, and that validate the approach. However, these are special-purpose, one-of-a-kind systems, designed to handle the particular needs of a single organization. Complementing these efforts is a growing amount of research and development aiming to develop general purpose DDBMSs. The SDD-1 developed by the Computer Corporation of America is an example (Rothnie *et al.* 1980). Such systems, like conventional DBMSs, are intended as off-the-shelf software packages capable of solving a wide range of database problems.

General-purpose DDBMSs fall into two important categories. The first consists of *homogeneous* systems in which all data is entered and accessed by the DDBMS. The second consists of *heterogeneous* systems which either integrate data already stored on existing DBMSs, or involve the use of relatively independent DBMSs at different network nodes. Potentially, these DBMSs can use different database schemas or even different data models. Most current research is directed toward homogeneous systems, since such research can concentrate on problems distinct to distributed DBMSs. The problems of heterogeneous systems are largely related to the problems of data mapping; these problems are involved and, as described earlier in the section on mapping, are just beginning to be addressed.

Specific research areas in DDBMS technology of special interest to DSS may be described as follows:

(1) *Distributed query processing.* This involves processing a query that accesses data at multiple sites of a distributed database. The first consideration is the new element of processing delay induced by communications delays between sites. The second consideration is the opportunity for parallel processing when several computers are involved in handling the query. Because of the very great variations in communications costs associated with various plausible query processing strategies, it is important to find techniques that will ensure satisfactory performance (Wong 1977).

(2) *Database design.* The major database design issue associated with distributed databases concerns the allocation of pieces

of the database to sites in the network and the degree of data replication that should exist. The design can have major effects on both the performance and the reliability of a distributed database system.

(3) *Heterogeneous databases.* A heterogeneous DDBMS must provide a unified view of data that has been stored on several different DBMSs, possibly with the use of different data models. Accordingly, the system must be capable of mapping the central view and the individual system views. The solution to this mapping problem will be difficult, but progress toward this goal implies improved capabilities for flexible use of different data with different programs.

The main area of interest of distributed database technology to DSS appear, however, to be the potential ability of such systems (and their underlying computer networks) to provide users at one site with access to other large databases, libraries, or software tools that may be useful in their DSS activities. Such sharing already takes place among research users of the Arpanet, as well as on commercial networks. Computer networks also facilitate communication among users located at different network sites. The mail facility of the Arpanet, which allows users of a given site to send electronic mail to users at other sites, is very widely used. This has enabled researchers to collaborate on papers without seeing or even speaking to one another, for example. Entire drafts of papers may be sent back and forth for comments and corrections. Integration of distributed database capabilities and computerized message services are being explored in the command and control context, both to enable users to interact on decisions based on data contained in geographically dispersed databases and to enable users to compose messages using the resources of several databases.

Database Hardware

Research aimed at the development of special-purpose hardware devices to assist in database processing has been actively pursued in recent years. Generically, a *database machine* is "any hardware, firmware, and software complex that concentrates or dedicates special purpose computing resources for supporting the database management portion of a computing system" (Yao *et al.* 1978). In theory, database machines can be free-standing, linked to main computers as special-purpose devices, or interconnected with other computers in a network. Two factors have helped to promote the development of database machines. The first is the improvements in LSI semiconductor technology, combined with reduced cost. The second is the growing demand placed on computing resources by increasingly complex DBMS components. The high-level database access languages and abstract data models described earlier currently require many levels of complex software for successful implementation, thus causing slow response. Moreover, increasing use of database technology involves the creation of larger databases, as applications require more and more complex retrieval and update capabilities. The secondary

storage structures (such as indexes) currently required to support such operations can become enormous for large databases, and search operations require considerable processing overhead. Finally, increasing demands for security and integrity protection functions for databases will require added computing power. Solution of these performance and complexity problems may not be possible without hardware assistance.

Yao *et al.* (1978) classify current developments in database machine technology into three categories: intelligent controllers and memories (Johnson 1975), special-purpose processors (Berra and Singhania 1976), and database computers (Baum *et al.* 1976, Canaday *et al.* 1974, Hakozaki *et al.* 1977, Madnick 1975, Marill and Stern 1975, and Schuster *et al.* 1978). Of these, the database computers are perhaps the most interesting, as they represent complete implementations of all the functions of a DBMS. Implementations of database computers have been simulated on conventional hardware dedicated to database processing (Canaday *et al.* 1974, and Marill and Stern 1975). Canaday *et al.* used the database computer as a back-end machine to a conventional computer. In the case of the Datacomputer developed by the Computer Corporation of America (Marill and Stern 1975), the computer is accessible as a specialized node on the Arpanet. In addition, a number of implementations of prototype database computers employing unconventional hardware are under development.

DSS shares the need for improved performance that prompted the development of database hardware. Database hardware might be used directly to address specific data-oriented requirements of a DSS, even though a DBMS per se might not be involved. Even more important, however, is the role that database hardware might play in providing support for advanced DBMS capabilities, which meet DSS requirements for flexibility, ease of use, and power, without sacrifice of necessary performance.

CONCLUSIONS

This paper has attempted to provide an overview of database capabilities potentially relevant to perceived DSS requirements. Although the treatment of both the DSS requirements and the database capabilities has been necessarily brief, it is hoped that sufficient flavor of these capabilities and their relationship to DSS requirements has been conveyed to promote further investigation. In the past decision support systems have appeared to stress adaptability to changing requirements, while database systems have appeared to stress preplanned integration of facilities that, once specified, have been relatively difficult to change. Given this inflexibility, it would not be surprising if the incorporation of database systems into decision support systems were difficult or inappropriate. However, as more and more data in organizations comes under the control of database systems, this situation will have to change. Much of the database research in recent years has involved attempts to improve the flexibility and range of applicability of database systems, in response to problems such as those addressed directly by decision support systems. It is to be hoped that the interaction of DSS and DBMS technology can be mutually beneficial.

REFERENCES

American National Standards Institute (ANSI) (1975) ANSI/X3/SPARC Study Group on Data Base Management Systems Interim Report. FDT 7(2). New York: ACM.

Astrahan, M.M.,*et al.* (1976) System R: Relational Approach to Database Management. ACM Transactions on Database Systems 1(2).

Bakkom, D.E., and J.A. Behymer (1975) Implementation of a Prototype Generalized File Translator. Proc. 1975 ACM SIGMOD Conference. Communications of the ACM 18(10).

Baum, R.I., D.K. Hsiao, and K. Kanan (1976) The Architecture of a Database Computer—Part I: Concepts and Capabilities. Technical Report OSU-CISRC-TR-76-1. Columbus, Ohio: Ohio State University.

Berra, P.B., and A.K. Singhania (1976) A Multiple Associative Memory Organization for Pipelining a Directory to a Very Large Data Base. Digest of Papers COMPCON 76 Spring. New York: Institute of Electrical and Electronic Engineers.

Birtwistle, G.M., *et al.* (1976) SIMULA BEGIN. Lund, Sweden: Studentlitteratur.

Blanning, R.W. (1979) The Functions of a Decision Support System. Information & Management 2(3).

Bonczek, R.H., and A.B. Whinston (1977) A Generalized Mapping Language for Network Data Structures. Information Systems 2(4).

Buneman, O.P., H.L. Morgan, and M.D. Zisman (1977) Display Facilities for DSS Support: The Daisy Approach. Data Base 8(3).

Buneman, O.P., and E.K. Clemons (1979) Efficiently Monitoring Relational Databases. ACM Transactions on Database Systems 4(3).

Canaday, R.H., R.D. Harrison, E.L. Ivie, J.L. Ryder, and L.A. Wehr (1974) A Back-End Computer for Database Management. Communications of the ACM 17(10): 575.

Carlson, E.D. (1979) An Approach for Designing Decision Support Systems. Data Base 10(3).

Chamberlin, D.D., M.M. Astrahan, K.P. Eswaran, P.P. Griffith, R.A. Lorie, J.W. Mehl, P. Reisner, and B.W. Wade (1976) SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control. IBM Journal of Research and Development 20(6): 560-575.

Champine, G.A. (1977) Six Approaches to Distributed Databases. Datamation 23(6).

Chen, P.P.S. (1976) The Entity Relationship Model—Toward a Unified View of Data. ACM Transactions on Database Systems 1(1).

CODASYL (1971) CODASYL Data Base Task Group April 1971 Report. New York: Association for Computing Machinery.

CODASYL (1978a) CODASYL COBOL Committee Journal of Development 1978. Material Data Management Branch, Department of Supply and Services, Canada.

CODASYL (1978b) CODASYL Data Description Language Committee Journal of Development 1978. Materiel Data Management Branch, Department of Supply and Services, Canada.

Codd, E.F. (1970) A Relational Model of Data for Large Shared Data Banks. Communications of the ACM 13(6).

Codd, E.F. (1971a) Relational Completeness of Data Base Sublanguages. Courant Computer Science Symposium Vol. 6: Data Base Systems. New York: Prentice-Hall.

Codd, E.F. (1971b) A Data Base Sublanguage Founded on the Relational Calculus. Proc. 1971 ACM-SIGFIDET Workshop, San Diego, California. November 1971.

Codd, E.F. (1978) RENDEZVOUS Version 1: An Experimental English-Language Query Formulation System for Casual Users of Relational Data Bases. San Jose, California: IBM Research Division, IBM Research Report RJ2144.

Codd, E.F. (1979) Extending the Data Base Relational Model to Capture More Meaning. ACM Transactions on Database Systems 4(4).

Computer Sciences Corporation (1977) MANAGE Data-Base Designers Reference. Report EOO253-01, September 1977.

Date, C.J. (1977) An Introduction to Database Systems. 2nd ed. Reading, Massachusetts: Addison-Wesley.

British Computer Society Data Dictionary Systems Working Party (1979) DDSWP Journal of Development. May 1979.

Foster, J.D. (1976) Distributive Processing for Banking. Datamation 22(7).

Fry, J.P., D.C.P. Smith, and R.W. Taylor (1972) An Approach to Stored Data Definition and Translation. Proc. ACM SIGFIDET Workshop on Data Description Access and Control. Denver, Colorado. November 1972.

Fry, J.P., R.L. Frank, and E.S. Hersey III (1972) A Developmental Model of Data Translation. Proc. ACM SIGFIDET Workshop on Data Description Access and Control. Denver, Colorado. November 1972.

Hakozaki, K., *et al.* (1977) A Conceptual Design of a Generalized Database Subsystem. Proc. Third International Conference on Very Large Databases. Tokyo, Japan. October 1977.

Hammer, M., and D. McLeod (1978) The Semantic Data Model: A Modelling Mechanism for Database Applications. Proc. 1978 SIGMOD Conference. Austin, Texas.

Harris, L.R. (1977) User Oriented Data Base Query with the ROBOT Natural Language Query System. Proc. Third International Conference on Very Large Databases. Tokyo, Japan. October 1977.

Haseman, W.D. (1977) GPLAN: An Operational DSS. Data Base 8(3).

Hendrix, G.G., *et al.* (1978) Developing a Natural Language Interface to Complex Data. ACM Transactions on Database Systems 3(2).

Herot, C.F. (1979) Spatial Management of Data. Technical Report CCA-79-24. Cambridge, Massachusetts: Computer Corporation of America. (To appear in ACM Transactions on Database Systems.)

Housel, B.C. (1976) A Unified Approach to Program and Data Conversion. Proc. Second International Conference on Very Large Databases. Brussels, Belgium. September 1976.

Johnson, C. (1975) IBM 3850 Mass Storage System. AFIPS Conference Proc., Vol. 44, NCC.

Joyce, J.D., and N.N. Oliver (1977) Impacts of a Relational Information System on Industrial Decisions. Data Base 8(3).

Kent, W. (1979) Limitations of Record-Based Information Models. ACM Transactions on Database Systems 4(1).

Madnick, S.E. (1975) INFOPLEX: Hierarchical Decomposition of a Large Information Management System Using a Microprocessor Complex. AFIPS Conference Proc., Vol. 44, NCC.

Marill, T., and D. Stern (1975) The Datacomputer—A Network Data Utility. AFIPS Conference Proc., Vol. 44, NCC.

McDonald, N. (1975) CUPID: A Graphics Oriented Facility for Support of Non-Programmer Interactions with a Data Base. Memorandum No. ERL-M563. Berkeley, California: University of California.

Morgan, H., *et al.* (1980) Database Management Systems, in What Can Be Automated? The Computer Science and Engineering Research Study (COSERS), edited by B.W. Arden. Cambridge, Massachusetts: MIT Press.

Nash, D.R. (1977) Building EIS, A Utility for Decisions. Data Base 8(3).

Nijssen, G.M., ed. (1976) Modelling in Data Base Management Systems. Amsterdam, The Netherlands: North-Holland.

Nijssen, G.M., ed. (1977) Architecture and Models in Data Base Management Systems. Amsterdam, The Netherlands: North-Holland.

Pliner, M., L. McGowan, and K. Spalding (1977) A Distributed Data Management System for Real-Time Applications. Proc. 1977 Berkeley Workshop on Distributed Data Management and Computer Networks. Berkeley, California. May 1977.

Ramirez, J. (1973) Automatic Generation of Data Conversion Programs Using A Data Description Language. Ph.D. dissertation. Moore School of Electrical Engineering, University of Pennsylvania.

Reisner, P. (1977) Use of Psychological Experimentation as an Aid to Development of a Query Language. IEEE Trans. on Software Engineering SE-3, 3.

Rothnie, J.B., and N. Goodman (1977) A Survey of Research and Development in Distributed Database Management. Proc. Third International Conference on Very Large Databases. Tokyo, Japan. October 1977.

Rothnie, J.B., *et al.* (1980) Introduction to a System for Distributed Databases (SDD-1). ACM Transactions on Database Systems 5(1).

Schneider, G.M., and E.J. Desantels (1975) Design of a File Translation Language for Networks. J. Information Systems 1(1).

Schuster, S.A., *et al.* (1978) RAP2: An Associative Processor for Data Bases. Proc. Computer Architecture Symposium. Blue Mountain Lake, New York. April 1978.

Senko, M.E. (1976a) DIAM II with FORAL LP: Making Pointed Queries with a Light Pen. Report RC 6034. Yorktown Heights, New York: IBM Research Laboratory.

Senko, M.E. (1976b) DIAM II: The Binary Infological Level and Its Database Language FORAL. ACM SIGPLAN Notices 11.

Shipman, D. (1979) The Functional Data Model and the Data Language DAPLEX. Technical Report CCA-79-16. Cambridge, Massachusetts: Computer Corporation of America. (To appear in ACM Transactions on Database Systems.)

Shneiderman, B. (1978a) Improving the Human Factors Aspect of Database Interactions. ACM Transactions on Database Systems 3(4).

Shneiderman, B. (1978b) A Framework for Automatic Conversion of Network Database Programs Under Schema Transformation, in Information Technology, edited by J. Moneta. New York: Elsevier/North-Holland.

Shopiro, J.E. (1979) Theseus—A Programming Language for Relational Databases. ACM Transactions on Database Systems 4(4).

Shu, N.C., *et al*. (1977) EXPRESS: A Data Extraction, Processing, and Restructuring System. ACM Transactions on Database Systems 2(2).

Sibley, E.H., ed. (1976) Database Management Systems. ACM Computing Surveys 8(1).

Smith, D.C.P. (1975) An Approach to Data Description and Conversion. Communications of the ACM 18(10).

Smith, J.M., and D.C.P. Smith (1977) Database Abstractions: Aggregation and Generalization. ACM Transactions on Database Systems 2(2).

Software House (1976) System 1022 Data Base Management System User's Reference Manual. Version 113, Rev. 3. September 1976.

Sperry Rand Corporation (1974) UNIVAC 1100 Series Data File Converter, Programmer Reference-UP-8070.

Sperry Rand Corporation (1977) Sperry Univac 1100 Series Query Language Processor (QLP 1100) User Reference. UP-8231 Rev. 1.

Stonebraker, M. (1974) High Level Integrity Assurance in Relational Data Base Management Systems. Memo ERL-M487. Berkeley, California: University of California.

Stonebraker, M., *et al*. (1976) The Design and Implementation of INGRES. ACM Transactions on Database Systems 1(3).

Su, S.Y.W., and B.J. Liu (1977) A Methodology of Application Program Analysis and Conversion Based on Database Semantics. Proc. 1977 SIGMOD Conference. Toronto, Canada.

Taylor, R.W. (1971) Generalized Data Base Management System Data Structures and Their Mapping to Physical Storage. Ph.D. dissertation. University of Michigan.

Thomas, J.C. (1977) Psychological Issues in Data Base Management. Proc. Third International Conference on Very Large Databases. Tokyo, Japan. October 1977.

Tsichritzis, D., and A. Klug, eds. (1977) The ANSI/X3/SPARC DBMS Framework Report of the Study Group on Database Management Systems. Montvale, New Jersey: AFIPS Press.

Winters, E.W., and A.F. Dickey (1976) A Business Application of Data Translation. Proc. 1976 ACM SIGMOD Conference.

Wiseman, T. (1977) Ambitious EFT: Quintet of Banks, Quartet of Vendors. Computerworld 11(23).

Wong, E. (1977) Retrieving Dispersed Data from SDD-1: A System for Distributed Databases. Proc. 1977 Berkeley Workshop on Distributed Data Management and Computer Networks. Berkeley, California. May 1977.

Yao, S.B., P.A. Bernstein, N. Goodman, S.A. Shuster, D.W. Shipman, and D.C.P. Smith (1978) Data-Base Systems. Computer (IEEE Computer Society). September 1978.

Zloof, M.M. (1975) Query by Example. AFIPS Conference Proc. 1975, NCC.

DOING AND SPEAKING IN THE OFFICE

Fernando Flores
Juan J. Ludlow
Stanford University

INTRODUCTION

Any discussion oriented to the design of communication and support tools for management must deal with the broader context in which managers operate and make decisions. The convergence of data processing, decision modeling, and word processing with computer communication networks in the new environment of computer-controlled devices in the office of the future must also be considered. The major concern for the 1980s will not be the merging of these various technologies—for this is already happening, together with the diffusion of electronic communication. What must be planned is how organizations will use the new technologies and how office and managerial practices will evolve under their influence.

In this paper we present the elements of a theory of communication and commitment. This theory can form the basis for designing new hybrids of computer and communications technologies. It is particularly relevant to evolving work in areas like Decision Support Systems because it sheds new light on the study of management practices and "decision making." The focus of the theory is a family of tools we propose to call Coordinators; they will be the central component of systems for supporting the activities of management and office workers.

The theory we have developed is fundamentally different from those that are commonly used to study communication and management. We adopt a unified approach in which communication is analyzed in terms of the issuance of commitments in conversations; the approach also takes into account that managers create, take care of, and initiate new commitments within organizations. This contrasts with the current tradition in which communication

is analyzed in terms of the transfer of information and management is equated with the making of decisions. We believe our new approach is much more closely related to the essential nature of management and communication. In the framework of our theory, organizations are institutional settings that orient the structure of commitments.

With this theory we challenge the received wisdom and cherished notions about design and communication. We wish to refer in particular to two erroneous notions.

The first is the common belief that design should be dominated by the desires of the *user*. Prejudices derived from this belief are (a) that the best way to discover what the user wants is by asking questions by means of interviews and by observing and studying the user, and (b) that the criteria for design will evolve inductively after a trial and error process. We believe that this approach is wrong. The structure of the interactions is not chosen or agreed upon by the users, but rather must reflect the structure of the deeds performed. The deeds are not independent of language and vice versa. The systems designer needs to understand this structure *before* starting to study and design; questions are important, but only if the researcher knows what to ask.

The second erroneous notion results from a poor understanding of communication. Although we agree with the consensus that communication is the crucial element in the office, we disagree with the notion that communication is merely the transmission of information or symbols. We challenge this concept, and consider it an entirely inadequate design foundation for the Office of the Future.

With these considerations in mind, the paper is divided into three sections:

— *Doing and speaking in the office*. Here we analyze—from our perspective of design—the nature of work in the office and present a preliminary discussion of the issues involved. We introduce the topic by seeking answers to questions like, What do people do in an office? and What do managers do?
— *Tools for the Office of the Future*. In this section a description of the new kinds of tools we envision is developed; we then discuss their feasibility and characteristics, with some preliminary thoughts on their impact on systems and on organizations.
— *Some final observations*. We conclude with some observations intended to clarify some potential misunderstandings that could otherwise arise from this work.

We would like to thank the following individuals for their valuable cooperation in the development of this paper: Bob Colten, Max Diaz, David Lowe, M. Laurentius Marais, Peter Stokoe, and Terry Winograd. They patiently went through the various drafts of this paper and pointed out to us, in vivid discussions, many mistakes and omissions. In any case, we take full responsibility for any errors remaining and for any misunderstandings that

may arise from the paper. Finally, we would like to thank the editors of these Proceedings for all their help in getting the paper completed on time.

THEORETICAL FOUNDATIONS

The intellectual framework for the design we propose for office tools comes from the general theory of communication and competence that we have previously elaborated (Flores). But, to help the reader and to give some autonomy to this work, we shall briefly look at the relationships between commitments and speaking, a central aspect of our theory.

Speech Acts and Commitments

In everyday speaking there are utterances like "I will see you tomorrow," where questions of truth or falsehood are not the primary concern, as the Oxford philosopher J.L. Austin pointed out some 20 years ago (Austin 1962). Depending on the context, this utterance could be an answer to a request or an expression of intention. In either case it is an expression of commitment to a future action; the primary concern seems to be whether the promise will be broken or the performance of the action will be unsatisfactory, from the perspective of the person to whom the commitment is made.

Austin called verbs like "promise," which are used to perform some act in normal speaking, *performative verbs*, and asserted that there were some 1,000 such verbs (or verb phrases) in English. The utterance of a performative in the present indicative brings forth, as an intentional act of the speaker, a reorganization of the social relations that make up the world of the participants in the conversation. This can be put more tersely by saying that in all such utterances, certain kinds of commitments are expressed.

Another fundamental insight stemming from Austin is that a kind of commitment is also expressed in utterances which have traditionally been analyzed as statements in terms of propositional logic and its variants. The utterance "It is raining now in Boston" in a conversation commits the speaker to provide, if so requested, evidence or reasons supporting his assertion. Otherwise the speaker risks not being taken seriously as a sincere believer of what he is saying. This is the kind of commitment that is normally made in assertions or affirmations. Of course, there is a whole spectrum of different degrees of commitment in such expressions, running the gamut from tentative guesses to definite assurances and testimony before a court.

The original insights of Austin were pursued further and developed, mainly by John Searle, into a theoretical corpus called the *Speech-Act Theory of Language*. In his book *Speech Acts*, Searle attempts a systematic account of the conditions or rules governing the performance of speech acts (Searle 1969). He introduces the essential distinction between illocutionary force

and propositional content. He uses the notion of *propositional content* to refer to the propositional act that is performed as part of a speech act. A propositional act never takes place alone; it is always expressed as a part of a speech act. The other component of a speech act is the *illocutionary force*—the expressed commitment of the speaker. In the previous example the illocutionary force is the commitment of the speaker to provide evidence, and the propositional content is the expression of the fact that it is raining in Boston.

Is it possible to find a framework of analysis that allows us to systematically generate these different kinds of commitments? Searle has attempted to answer this question; in our opinion his research has been very successful in distinguishing the basic *dimensions* of the space of social commitments generated in and through linguistic acts. Searle distinguishes five kinds or 'families' of illocutionary acts along these dimensions of commitment,* namely assertives, directives, commissives, declaratives, and expressives.

We shall give a short account of the classes of directives and commissives here, and later we shall add some comments about other kinds of commitments.

Directives

The distinguishing characteristic of a directive is the commitment by the speaker, as he asks for the (future) performance of some action by the hearer. The propositional content of the directive expresses the action to be performed.

As a serious person issues a directive he is also expressing a desire that an action be performed. It is clear that the utterance "I request that you go to our headquarters to give a lecture" also involves the commitment, "I would like you to go to headquarters" or "I have the desire that you go to headquarters." However, the converse is not necessarily true; this is apparent in the following example: "I would like you to go to Rome, but it is necessary that you stay here because there is something more important for you to do here." An imperative, where no illocutionary verb is explicitly mentioned, is normally understood as a directive. We may also distinguish actions such as requests, petitions, implorations, and orders as members of this class. Questions are also a kind of directive, in which the act that is requested is itself a speech act—an answer to the question.

The phenomenon of temporality is a general feature of the expressions in this class, insofar as an action to be performed in the future is specified. This feature is much more important than is apparent at first sight. The hearer always needs to know when and where the speaker wishes a requested action to occur.

---

*Searle introduces the notion of *illocutionary point* as the basic criterion for the classification of speech acts.

Of course, in many cases this is obvious and it does not appear explicitly in the spoken utterance. At other times it is open to some interpretation. Finally, there could be a certain vagueness that may potentially give rise to misbehavior.

Commissives

Searle gives the name *commissives* to the particular class of acts in which the speaker becomes committed to the future performance of an action. As a speaker utters a commissive—a promise for example—he is also making the commitment that he has a serious intention to perform the action. For example, in the utterance "I promise to go to London," an obligation is generated and an intention is expressed as part of the commitment.

Verbs such as *swear*, *commit*, *vow*, and *pledge* in the present indicative are normally taken as commitments of this kind. Commissives also have the remarkable feature, already pointed out for directives, that they bring about a reorganization of the world through the creation of a concern for a future act. Both kinds of speech act have as their *illocutionary point* "to get the world to match the words"—thus Searle's characterization of them as having "World-to-Word" direction of fit. This contrasts clearly with assertives, which have the opposite "Word-to-World" direction of fit. That is, assertives have as part of their illocutionary point "to get the words to match the world."

Some Comments on Commitment

All utterances are commitments according to this theory. In everyday use the word *commitment* is normally associated with what we call commissives, but this is an understanding we challenge. We instead assert:

> It is unavoidable that commitments are expressed and *listened to* by the participants in a conversation.

What is peculiar about commissives is the double self-referentiality of the commitment of the speaker, i.e., the expressed commitment to the *intention* to perform the act and the creation of the obligation to perform the act, as such.

Another interesting feature of human language and action that bears comment here is found in expressions such as "You shall not kill." For these kinds of expressions, saying that one did not know about them or that one had never made the corresponding promise is not generally an acceptable excuse. One is expected to act as if one had in fact made such a promise. In everyday language we call these kinds of expressions *obligations*, *customs*, *mores*, etc. These expressions can have different degrees of strength.

On the other hand we must emphasize that there is no one-to-one correspondence between commitments, as kinds of acts, and

English verbs. Furthermore, many acts can be realized by a complex verbal expression or, conversely, some verbs may simultaneously have two or more dimensions of commitment; for instance, promulgation of a law is ordinarily a declarative and a directive.

Summary

The points made above may be restated in the form of four succinct assertions.

Assertion 1: The units of analysis of communication are the *commitments* that a speaker makes by speaking or, equivalently, by performing acts such as writing, gesticulating, or even by an interpreted silence.

Assertion 2: These commitments may be decomposed into two components—the *illocutionary force* and the *propositional content*, using Searle's terminology for speech acts.

Assertion 3: Illocutionary forces may be grouped into five classes, according to the different kinds of social obligations to which the speaker commits himself by speaking.

Assertion 4: Although the speaker of a language like English may express illocutionary forces in many different ways, *intelligible* communication occurs when both speaker and hearer have listened to the same illocutionary force and to the same propositional content. To state this more precisely, both partners in a conversation are able to come to an agreement if claims arise about the intelligibility of their utterances, and these claims are always about the illocutionary force and the propositional content.

DOING AND SPEAKING IN THE OFFICE

In this section we shall develop answers to the questions *What do people do in an office?* and *What do managers do?* But, before doing so, we shall make a few comments on the nature of questioning and answering.

Answering a Question

In order to answer a question, we generally need to know what the conditions of satisfaction are for giving "an acceptable answer" to the person asking the question. But the question itself is asked because a breaking-down, an interruption, of our normal activity and way of life has occurred. If we want to explore the background of a question we need to ask, *What are the sources of concern? What are the problems? Where will the potential solutions come from? What new entities are revealed by the question?*

Following the paths opened by these questions directed to the original question, we can explore and clarify some possible answers. It is not necessary that we always arrive at an answer. Very often we just walk away with a better understanding of what we have been asked, i.e., we have developed an interpretation of the situation that triggered the question. If someone who is presumed to be a master of the field asks, *What is Law?* the value of the inquiry will not end in a simple uncontroversial answer; it will be the beginning of a reorganization of the whole prevailing way of dealing with the subject. This is what is meant by "a better understanding." If the same question is asked during an examination for a university course, the question has been oriented by what the professor considers to be a valid answer.

Our original question, *What do people do in an office?* stems from our concern with design; therefore we expect our answers to help in the design of new tools. We are interested in producing some directives for dealing with recurrent patterns of break-downs. In this way we hope to improve the efficiency and the way of life in offices for both workers and managers. The directives are not simply a certain kind of utterance. They imply a reorganization of understanding—a new way of speaking, new actions, new entities, and new workers. We use the word *reorganization* because a new understanding is always a new interpretation—a recreation of previously existing approaches, with the existing language and style embodied in an old generation of practitioners.

What Do People Do in an Office?

The normal answer to the above question may be, "Some secretaries type, some people answer the phone, the engineer is doing some calculations, the salesman is selling, and the manager is discussing with someone the need to hire a new employee." Other respondents, using modern jargon, may say, "Here we process information (we have a lot of machines and data bases), and our managers make decisions."

If the purpose of these answers is to satisfy the curiosity of some occasional foreign visitor, they may be considered satisfactory. However, if we want to participate in a discussion about buying a new machine, a copier for example, the answer to the same question could be, "The people in this office make 5,000 copies per day, with peak periods of 10,000 copies." This answer is useful because it describes what people do in the office; it may help to make the decision about the copier and to issue subsequent orders. This type of answer allows one to pursue successful conversations as an effective way of dealing with the situation.

We may now repeat the question, What do people do in an office? in the context of the office of the eighties in America. Before attempting to give an answer we need to explore some of its historical background. There is sufficient evidence and overall consensus that the principal issue concerning the Office of the Future is that productivity is decaying in America and that something needs to be done about it. The "office of the

future" is proposed as a solution to the problem of low productivity but no one bothers to define what the term means. Nevertheless, new possibilities and new problems are emerging as a result of technological revolutions in the semiconductor and electronic communication fields. We may ask, How will experts in these new fields provide answers to our question?

On one hand, the old answer would have been that these experts will provide more efficient tools to process information and make decisions, and that the crucial tools to be designed are called "Management Information Systems," based on scientific systems analysis (Morgan and Soden 1973). On the other hand, the unanimous answer to the new experts is that what people do is, obviously, communicate (Rogers *et al.* 1976). But when we begin to ask what they understand as *communication*, the unanimity disappears; many of the "new" answers are not really different from the old answer, except that they involve new devices emerging from the technical and economic possibilities presented by computers and communications (Bair 1978).

Our Answer

If we examine the basic issues underlying the questions, "What do people do in an office?" and "What is communication in an office?" we find that the questions are not truly different. Our theory of commitments and conversations allows us to give an answer to these questions that provides guidelines for examining the work in an office or organization.

Let us use the insights gained into the relationship between commitments and action to analyze organizations. For this purpose we make the following assertions:

> Organizations exist as networks of directives and commissives.*
>
> Break-downs will inevitably occur and organizations need to be prepared for them. In the process of coping with break-downs, whole new networks of directives and commissives are triggered.**

---

*These directives include orders, requests, consultations, or offers; commissives include promises, acceptances, denial.

**The notion of *breaking-down* was brought to the center of the philosophical discussion by Martin Heidegger in his book *Being and Time*. Heidegger points out how the world, as a successful transparent being, is brought forth by a disturbance as a concern with the perturbation. He also describes how the plexus of referentiality and proximal commitments appears in the break-down, as practical know-how in coping with the perturbations. Heidegger illustrates his presentation with the example of someone working with a hammer. Coping with equipment and things is called by Heidegger *concern*. Dealing with people is called *solicitude*. The whole structure of coping with the World is called [continued]

> The process of division of labor may be considered a cultural heritage of ways to cope successfully with anticipated break-downs. This has been a constant concern for managers.

Let us present an example that illustrates these three assertions. Consider a company that sells books by mail. Requests for books arrive in letters worded in the following way: "Please dispatch to me, and charge to my account, the book XXXX, by the author such and such." If the business is well organized the client will be attended to quickly and the company will make some profit. When the number of such requests is large, several persons will carry out different parts of the work in the organization. Some will receive the requests, others will dispatch the books and invoices, send letters of acknowledgment, order books from the publishers, anticipate the levels of the inventories, and so on.

The organization must be careful to influence the orders from the clients through advertising and promotions; it would be fatal to receive hundreds of requests for cookbooks if the company specialized in scientific books.

Break-downs are the way in which concerns appear to each member of the organization. Many are already anticipated in the form of work specialization, standard forms, rules for credit, policies about levels of inventories, etc. In this sense the notion of breaking-down also applies to daily notions such as negative perturbation or trouble. To be in business is to know how to deal with break-downs, to anticipate their arrival, and to face them. The concerns are concerns about doing something. The two *par excellence* social forms of concern are contracts, two mutual promises, and the internal managerial conversations within the organization. The principal flow in the network of action is generated by the contractual relations of the organization, linked to the internal managerial relationships. Internal conversations may be supported by other networks of action that appear to be related only in a very indirect way with the principal flow.

Janitorial services for a shop provide an example of this case. The janitors come every day at a certain hour. Their

---

care. Heidegger describes the structure of *concern* that is manifested in the structure of referentiality, which is in turn revealed by the prepositions of language. However, he lacks a rich framework to speak about *solicitude* with the same precision; in order to do this it is necessary to have an interpretation and a framework for referring to social relationships, such as the one provided by the speech-acts theory as developed by Austin and Searle. One of our purposes in this and other works has been to pursue further the study of the structure of social connectivity, with the aim of discovering the minimal communicative tools that one must have in every moment of interaction. We have produced our synthesis between these two different and apparently conflicting philosophical traditions with this design aim in mind.

actions have been defined in a contract, in terms of hours per day, rates per hour, fines for delays and absences, and so on. Specification of the products that the organization sells is not part of the language in which the janitorial contract is written. However, it may happen that someone connects the cause of claims about the low quality of certain products to the janitorial work. Then the connection of these two networks of conversations is revealed.

Our answer to the question, *What do people do in an office?* becomes obvious in this context. People issue utterances, by speaking or writing, to develop the conversations required in the organizational network.

In our initial formulation we insisted that 'Directives' and 'Commissives' were a shorter way to grasp a whole gamut of more subtle distinctions. As a matter of fact, we may develop finer criteria of analysis that include acts such as offers, invitations to offer, requests about products, invitations, questions, orders, instructions, and general regulations as a subset of Directives. In the same way authorizations, approvals, appointments, contracts, etc., would be considered 'Commissives.'

Are there also other kinds of acts? Of course, assertives, expressives, declaratives, and the performance of actions are just as essential. But to grasp the basic dynamics of the office as a network of commitments, the emphasis on Directives and Commissives seems convenient. Still, a few comments about the other kinds of acts are now necessary.

Assertions. This is the kind of utterance in which a speaker becomes committed that the belief (or disbelief) that is expressed is justified and justifiable. In other words, the speaker is committed that something is the case, to the truth of the expressed proposition.

There are different kinds of assertions, such as assurances, reports, evaluations, and predictions. Their central feature is the speaker's implicit offer to provide some evidence or grounding for what he is asserting. If he fails to provide this, it could be said that the speaker has made a false assertion. If this failure is intentional the assertion will be called a lie. For us, assertions arise in conversations as answers to questions in a context of action; i.e., assertions are actions in a conversation stimulated by a listened-to request. It is within these particular situations that an assertion will satisfy a validity claim.

Declaratives. The distinguishing feature of the declarative form of commitment may be described as follows: when it is performed by a speaker, it brings about a correspondence between the propositional content and the World. Successful performance of these

> utterances guarantees that the propositional content corresponds to the World. Through declaratives it is possible to bring about a reorganization of the whole social space, including the rights of the other members of the universe of discourse to question what is declared as something true or appropriate.
>
> Declarative forms of commitment occur within an organization in a very particular way: they are executed with the support and within the protection of institutions of Law. When someone is appointed to a new position, or receives the power to sign documents like bank checks on behalf of a company, it is necessary that this act of transference of a new status of power be carried out by people especially endowed with the authority to perform it; as well, this act must be done in the proper circumstances. Formal organizations, such as corporations, trade unions, partnerships, government bureaus, and churches, have been delegated—through some *constitutive act* of an authority—a set of special directives or rules, endowing the particular organization with a space to exist and with certain rights and obligations. In accordance with this constitutive act of law, validity is given to the appointment of people, enactment of internal rules, issuance of orders and commands, etc. In the same way, the process of modifying some of the rules and norms must refer to the constitutive act and to the preexistence of other normative institutions of law and social institutions. The articulation of the normative space, as a network of declaratives and directives, supports the validity of many of the acts of the organization. Therefore, by performing an analysis of declaratives, it is possible to introduce preventive steps to insure the validity of many of the actions that occur in actual conversations.
>
> A more complete account of the use of declaratives in private acts, such as the use of language itself, and the introduction of definitions of different kinds, like those of mathematics, will not be pursued here.

For the purposes of this work we consider that both assertions and declarations appear when some previous directive act has triggered the circumstances of interaction, which can be satisfied by one of these acts. To keep this paper limited in size and scope we shall not discuss expressives, the remaining class of illocutionary acts. Finally, the discussion of actual performance will be deferred to a later section, where it will be discussed in the context of our model for conversations.

Returning now to our initial question, "What do people do in an office?" we would like to review our answer, namely, "People in an office participate in the creation and maintenance of a process of communication. At the core of this process is the performance

of linguistic acts that bring forth different kinds of commitments."

We are not asserting that people are aware of what they are doing; they are simply working and speaking, more or less blind to the pervasiveness of a commitment's essential dimensions. It is because of this peculiar fact that one of our recommendations for organizational design takes the following form:

> The process of communication should be designed to bring with it a major awareness about the occurrence of *commitments*. Every member's knowledge about his participation in the network of commitment must be reinforced and developed.

We now have a new framework with which to discuss organizations and the activity of designing them, as well as basic guidelines for designing office tools. To summarize this we add another three assertions to those already presented earlier in this paper.

> *Assertion 5:* A background of obviousness and relevance is always part of listening, listening that always goes beyond the spelled-out words that the speaker and the hearer speak and hear; in this context the propositional content of a speech act cannot be seen as a symbolic representation of some external reality that exists independently of language: the speech act imparts *what is not obvious*.
>
> *Assertion 6:* Every interaction is triggered by a listened-to utterance with a directive dimension. During a conversation possibilities open for a next step by the speaker or the hearer; these can be characterized at the level of generality of the illocutionary forces, i.e., in terms of the different kinds of commitments that are expressed in speaking.
>
> *Assertion 7:* The speaker and the hearer are always able to identify their actions in a spatial-temporal framework. That is, they can always answer the questions, *Where? When?*

The seven assertions constitute our attempt to summarize a line of research in the Philosophy of Management and Language that we have been developing. They provide the foundation of our approach for the design of new tools for the Office of the Future. They are not rigid principles of design; they are modest efforts at a synthesis intended to embrace the most important elements of a new discourse. For a more elaborate philosophical discussion we refer the interested reader to other works (Flores); in the remaining sections of this paper we shall continue our examination of the activities of office workers and managers.

What Do Managers Do?

It has been remarked by several observers (see Mintzberg 1973) that the activities of managers are not well represented by the stereotype of a reflecting solitary mind studying complex alternatives—as is presupposed when decision making is identified with choosing. Instead, managers appear to be absorbed in many short interactions, most lasting between two and twenty minutes. Managers manifest a great preference for oral communication—by telephone or face-to-face. The language used by various researchers to describe these interactions differs, sometimes illuminating and sometimes hiding different aspects of the experience. Some descriptions say that the managers make decisions, and others say that they process and disseminate information.

Resolution Versus Decision Making

A commonsensical, somewhat obvious, observation is that decision-making activity is important, but it is not clear what this activity is and how it is executed. If someone hires a new employee or signs a new contract, one can certainly say that a *resolution has been reached.* We prefer to use this expression instead of "a decision has been made," because decision making has been identified so much with the choosing of alternatives. Choosing alternatives is a particular case within this broader process of resolution. The crucial point is that when a resolution has been reached, different commitments can be expressed. In an organization the most important resolutions are those which imply some *Directives*, or *Commissives* with a *Declarative* dimension.

The major part of this process of resolution is something that we can articulate in terms of a network of conversations in which many actors may participate. Sometimes we can specify the conditions of further inquiry needed to attain a resolution. On other occasions the process will come from a more prolonged hesitation and/or debate. Only in some cases will the phenomenon of choosing between alternatives occur; a process of ranking according to some metric or other criterion may occur even less frequently. A narrow identification of resolution with the choosing of alternatives has been a permanent blindness of Management Sciences.

It is not the purpose of this paper to explore the fascinating paths opened by a fresher, more analytical perspective of the process of resolution. But there are two aspects of the previous discussion that we find extremely relevant for our purposes, namely, the notions of commitment and conversations.

The notion of *commitment* has been fundamental to the discussion developed in this paper. The term *conversation* was present in latent ways as we adopted the perspective of looking at organizations in terms of networks of Directives and Commissives. The notion of conversation allows us to focus a little more closely on the phenomena of interaction.

An Example of Conversational Analysis

At this point it seems appropriate to introduce an example to clarify our notion of conversation. Let us consider a common situation in an office, a conversation between a manager, M, and a subordinate manager, S.

M: I need your cost estimate of the Johnson construction bid early next Friday.

S: OK, you'll have it at noon. Is that all right?

M: Yes.

Our analysis of conversation allows us to identify the first utterance as equivalent to "I order you to give me your cost estimate for the Johnson construction bid early next Friday." We can identify its illocutionary force as an Order belonging to the class of Directives. The propositional content is an action requiring the subordinate to give the manager details about the conditions of the bid. The "OK" that follows has, first of all, the function of recognition that what was listened-to, heard, and understood was sufficient. "Early next Friday" is the condition of time, a general requirement of any action to be performed as a result of a Directive. The place is not uttered explicitly, but it is understood as somewhere obvious—the office of the manager. The simple "OK" is a promise that can be analyzed in the following terms: "I promise that I will give you the cost estimate you want." The next utterance may be understood as a polite request for more precision, for the manager only indicated somewhat vaguely "early next Friday."

Another possibility is to analyze this step as a counter-offer, implying a polite decline to have the cost estimate ready by 8 o'clock in the morning. For us the important point is that two commitments have been made at the end of the agreement. The manager has received the promise about the report, and he has agreed (promised) to receive the cost estimate by noon next Friday. Two new concerns have been brought forth in the life of both persons.

We must stress that these problems of analysis occur only because we are not the actual speakers. There are usually no ambiguities in what the speakers intended to say, nor do they need to ponder, as we do, the significance of what they are doing. Still, this analysis is relevant to the domain of design.

We can extend this analysis with some modifications when the commencement of the interaction occurs as a result of a petition to a peer or to a superior, as the result of an offer received from a supplier, or as a result of a letter of invitation offering a product to a potential client. In all these cases, we have the same phenomena—a number of interactions are triggered, compelling us to attend, decline, accept, counter-offer, ask for clarifications, or answer some questions. We can call all of these concerns about potential commitments for future action.

We propose to call this succession of states, starting with a directive and ending with some performance or a definitive rejection, a *conversation for action*, or simply a *conversation*.

Conversation as Selection of Possibilities

This is not the place to pursue the analysis of conversations. What is important here is to grasp intuitively that the development of a conversation requires a certain finite selection of possibilities that are defined by the opening Directive and the actual conduct of the speaker. It is like a dance, giving some initiative to each partner in a specific sequence. It is also important to observe that from the design point of view a language for dealing with all these different situations may be specified, using a small number of distinctions; this will provide us with the capability to design very powerful tools.

Our Answer to the Question "What Do Managers Do?"

We may say in a simple way that managers create commitments in their world, take care of commitments, and initiate new commitments within the organization. But, in our statement, *world* is also what is brought forth through language as a commitment established by an utterance. In other words, managers engage in conversations.

The idea that the activity of managers has to do with the expression and articulation of commitments seems to be a simplistic answer. In fact there are many other things that managers do. For example, they negotiate, they speak in public, they are entrepreneurs. How would we characterize these activities? We may answer in the following way. Of course managers do a lot of things, but when you reduce all the acts to their basic dimensions, you will arrive at the kinds of commitments that we have been studying. But there are some deeper questions that merit attention. We may ask, What kind of relationship exists between our commitment language for descriptions and the other kinds of descriptions? What kinds of design, instructional practices, etc., can be better illuminated by these different languages? These important questions are not pursued in this paper.

There are other important issues worthy of mention. The first is that managers, like all other people, *fall* into conversations. This refers to the fact that we tend to erase the rest of the world when we are in conversation, and that we need to take special care to remain aware of it. However, if we maintain that awareness we run the risk of being distracted from the actual conversation. The second point is that many conversations may start with writing or the use of other media that imprint the conversations with special characteristics. If the conversation is in a letter, we become dependent on the efficiency of the mail service and on distance. If we call a person, his phone may be busy or he may not be there. If the medium is not verbal, we need many additional features to confirm that we are still carrying on

the conversation. These features are important because, as we shall see later, the interpretation of silence is an important matter.

TOOLS FOR THE OFFICE OF THE FUTURE

In this section we attempt to provide answers to our previous questions about the design of tools for managers and office workers.

What managers need are tools to produce illocutionary acts and to simultaneously provide a collection of functions for dealing with the complexity, the absorption, and the unrelenting pace of the commitments. At the same time, other members of the organization need to be connected through extensions of the same tools to prepare their own utterances. Through these tools the managers and the other members of the organization will maintain a better awareness of their activities, improving their effectiveness and the quality of their interactions.

The permanent headache of the managers is the continual necessity to deal with questions like the following:

— What is missing?
— What needs to be done?
— Where do I stand in terms of my obligations and opportunities?
— How do I communicate what I want to say?
— How do I avoid being overwhelmed by data and obvious details while maintaining accessibility and control?

At this point an initial answer to some of these questions can be given in terms of functional capabilities like the following:

— What are the pending requests (Directives) to me?
— What are the pending requests (Directives) from me?
— What are the overdue promises given to me?
— What are the overdue promises made by me to others?

Of course, it will only be possible to have monitoring capabilities for requests, promises, and offers if they have been previously recorded. However, this can take place at the moment when the issuance of the communicative act occurs.

The connectivity of the whole organization may be treated in a similar way. Filters can be designed to increase the transparency of the mutual interactions, in order to avoid pollution by excessive, inopportune *junk mail* that invades the whole organization in the form of unwanted interactions. At the same time, significant communications can be speeded up.

Indicators of failure can be provided automatically and can be investigated routinely as part of the organizational analysis. These are some of the principal additional tools that are derived from our approach to management and communication. Although there

are others, we are not going to pursue them in this paper. What we are going to do is to discuss in more detail the notion of conversation from the perspective of the design of new tools.

The kinds of new tools that we visualize are devices belonging to the new digital technology of communication; they may be embodied within a physical entity such as a modern PBX,* connected to a hybrid device combining a telephone and computer terminal. By using these devices, managers may interact and transparently issue the proper utterances. At the same time the manager can ask questions about his other commitments, losing this transparency to a minimal degree. In other words, the devices must be accessible for monitoring purposes, in addition to being the vehicles of inscription, and of transmission of the inscription.

There is no doubt that a great transformation is taking place, driven by technological innovations and the new products offered by the computer industry. Energy shortages, declines in productivity, and the growing cost of salaries contribute to the creation of a new scenario for the office of the next decade. The crucial problem will be one of organization and design. How can we design equipment, computer networks, and communication networks to be effective tools, to cope with the difficulties? How is it possible to avoid being swamped by a legion of new technological gadgets that do not improve life or human efficiency? These are difficult questions, so we are not going to deal with them directly.

Initially we want to address only two questions:

> Is it possible to design new kinds of products aimed to help managers to communicate effectively? This will be called the question of *feasibility*.
>
> Can we characterize some of the features of these products? This will be called the question of *characterization*.

At the end of the chapter, we deal with some other implications for the design of organizations and systems. We will conclude by presenting some caveats with respect to computer communications for organizations.

The Question of Feasibility

Our answer to the question of feasibility is strongly positive, supported by the theoretical analysis presented in the previous sections and in other works. Taking the risk of being repetitive, we will review our principal findings, using our new language.

---

*Private Branch Telephone Exchange

An organization develops its activity through the expression of commitments, through the articulation of commitments among its members, and in relation to its external commitments. These commitments form the deep structure under the surface of linguistic communication. In other words, they are the basic units of analysis for linguistic interactions. The dynamics of these interactions and the issuance of commitments can be reduced to a finite number of possibilities of states and issuances.

Based on these considerations, it is possible to develop the capacity to build a collection of tools that are easy to learn and easy to use. In a very important way, these tools will provide an *intelligent answer* to the question, *What do I need to do?* We know that this question can be interpreted as: *What are the states of my engaged commitments? How are they going?* The tools will provide simple but rich capabilities to monitor and issue commitments.

At this point it is reasonable to pose the question, Why will managers and office workers want to use these tools instead of accomplishing their tasks using more traditional means, such as face to face contacts, mail, or telephone? We are not saying that the use of these tools will be imperative. Face to face contacts will continue to occur on many occasions; they are necessary and convenient. However, we need to recognize the current importance of distance as a factor and a barrier in human work. For this reason we need to use other means of communication, such as mail or the telephone. Even in these cases, it may be necessary to continue a conversation, and therefore records and reminders are required. Although tools to help with these problems have existed for a long time, it is not clear that they are convenient for the present situation.

We are all familiar with the annoying experience of using the phone and not finding the person we want. Statistics indicate that the probability of contact in this situation is less than 30%. There is also a need for filtering mechanisms to protect people from interruptions during moments of absorption in certain activities.

Mail is not very effective for quick coordination, because the rhythm of a conversation is constrained by the scheduling of collection, classification, and delivery. However, mail is relatively cheap and it enables one to reach people nationwide and worldwide. The telephone is certainly better if a call is connected, but it is relatively more expensive. The benefit or opportunity costs go beyond considerations of design. How much the partners of a conversation are willing to spend to communicate by different means must be determined for a broader analysis. The notions of conversations and networks of secondary and coordination conversations will provide us with the basis for making such an analysis.

The proponents of electronic mail have heralded its appearance as a solution to many problems, citing the new possibilities it brings, such as an easy interface to data bases and text

editing functions. However, the emergence of these new technologies presents new problems. The principal problem is the proliferation of unwanted interaction. People already living in the world of electronic mail are experiencing a new form of *junk mail*. Electronic mail makes it very easy to bring new entities into existence and to initiate new interactions. Subtle mechanisms designed to control access begin to disintegrate, making it necessary to return to the mechanisms of privacy and hierarchical control. However, these in turn introduce additional costs in the form of underutilized systems. Another possible solution is to make the organizational hierarchy evident through the structure of protocols for communication. Unfortunately this has bad consequences for the morale of the organization.

Other problems stem from the nature of managerial conversations. In general, managerial conversations are short and intermittent. They are usually free form, in the sense that they are not part of the rule-governed conversations of an organization. Managers need new ways to interact; they need new communication tools with simple and clear commands. They do not need a complex keyboard. They are neither scientists, typists, nor computer professionals.

We may ask as a final question, *Aren't we proposing a nontransparent solution to the problem of processing meaning by the methods of processing language*, as investigated in the fields of Artificial Intelligence (AI) and Language Understanding? We can answer this question with a very emphatic assertion. The fact that there is funding and researchers are working in the directions proposed by practitioners of Artificial Intelligence does not predict future success. If someone believes in such an argument, he is digressing from the principles of scientific discourse. We have serious suspicions that many of the practitioners of AI are not even dealing with language; rather they are dealing with a derivative phenomenon generated by language, which emerges as a consequence of the inscription appearing as the written word. If we assert this, we are not saying that all they do is useless, unreal, or wrong. We are only emphasizing that the core of language and communication is not being grasped. Practitioners of AI can produce nice results in the domains of computer symbolic or algebraic manipulation and thereby extend human capabilities.

We have insisted that the unity of human communication derives from the making of commitments by means of the issuances of utterances; these are events that take place in contexts. We assert that it is impossible to find any kind of formal correspondence between the conventions of the written word and the structure of commitments in a conversation. It is not possible to reconstruct their structure by processing the written word. This is the case because commitments do not belong to the domain of inscriptions, but rather to the domain of human interaction. We propose to make the user aware of some of the features of this structure and at the same time to provide him with the tools that can couple him with the different steps of interactions.

The price the user will have to pay is the following: he will be subjected to a certain number of possibilities that are, necessarily, more restricted than those of the spoken word. But this is not necessarily bad. It is something that is always happening. We are always subject to constraints imposed by the nature of the written word and transmission media. Letters follow certain conventions, commercial letters and telephone conversations additional ones. The crucial point is to ask whether the additional conventions are ergonomical, viable, and economical. 'Ergonomic' refers to the possibility of incorporating extra conventions into a normal individual's skilled behavior after simple training. Economic viability refers to the necessity that the new costs are compensated for by gains in efficacy. These are points that we view with optimism, but they will not be debated here. Our concerns turn rather to the technological functions of Coordinators.

The Question of Characterization

We shall now characterize the class of products/tools that we call Coordinators. We shall try to point to the basic elements that clearly distinguish these products from other similar products. We shall not discuss the specific technological features and implementation of the tools. The intention of this description is to convey the ability to generate a concrete product, given specific technological constraints.

We are designing communications tools. These tools must be embedded in a physical transmission system of some sort, such as the switched telephone system, a computer network, or a timeshared computer.* This basic requirement often stems from the nature of the modern corporation. Modern corporations and governments are widespread, spanning many buildings, cities, and even countries. Under these circumstances the need for reliable, secure, and efficient communication makes the construction of communication networks a fundamental concern. The Office of the Future will develop and grow in this context. It has been our contention in this paper that the problem of coordination of action becomes crucial in these situations. For many organizations it is even a matter of survival. Furthermore, the complexity of organizations can no longer be controlled without the appropriate tools. It is clear what this means in the context of this paper: the networks of commitments and the conversations in which people are participating are becoming larger and more complex.

In the context of these observations, the Coordinator belongs to a class of products designed to construct and control conversation networks in large electronic communication systems.

---

*The switched telephone network with micro-computers can be the basis for such tools. It supplies all the required hardware technology.

Illocutionary forces are the basic concept introduced in this paper. We have said that these forces are the building blocks with which conversations are constructed, and with which conversational networks are in turn built. The power of these forces is that they correspond to the different kinds of commitments that people make while carrying out their daily activities. The crucial point is that behind all activity there is at least one pair of commitments: one by the requestor and the other by the requestee. Coordinators manipulate the illocutionary forces as their basic units. People interact in conversations and coordinate their actions through the issuance of these illocutionary forces. We can say this because illocutionary forces basically link commitments with action—human action—within a framework of time. Time is the crucial element if coordination is to be achieved. The Coordinator will offer simple and yet powerful ways to issue these illocutionary forces with their associated features—namely, time, place, and propositional content. It is important to note here that time, place, actors, and commitments are required elements from the point of view of coordination.

For certain applications we can introduce new distinctions at the level of specific illocutionary forces. For example, we may want to distinguish '*Request for budget*' from '*Request for sick-leave.*' Clearly these two requests have in common all the particular distinguishing features of a request; but they also have some additional features that depend on the particular organization in which they are issued and legal context that regulated them. When one discovers a recurrence of certain types of conversations, one can analyze the conditions for the successful performance of the speech acts of a given conversation. One can then incorporate these conditions in the Coordinator at the level of issuance of the speech acts.

To make this presentation more concrete, we present a scenario in which a Coordinator has been implemented. Let us assume that the basic hardware substate—i.e., the transmission, storage, and control mediums—is provided by a timesharing system to which a number of stations are attached. These stations give the members of the organization access to the services provided by the Coordinator. These stations take the form of display terminals, with special keyboards. The keyboard is divided into three main areas, one for the normal alphanumeric keyboard, one for keys that correspond to the main illocutionary forces that have been distinguished, and the remaining area with keys to enter dates and times.* By making this distinction in the keyboard organization we want to point out that we must make available to the users of these stations efficient ways to signal the illocutionary act that they want to perform.** In summary, users must have good ways to mark the illocutionary forces and to inscribe the propositional contents of their utterances.

---

*Clearly this is only one of various choices; another would perhaps include a pointing device, and so on.

**Another way to achieve this is by the intelligent use of menus.

Notes about the Design of Systems and Organizations

The practical consequence of the above discussion for the design of organizations may be described as follows. It is possible to have an observational theory that precedes the empirical study of organizations. It is beginning to be recognized that the power of empirically-based science is rooted in the power of its idealized "pre-anticipations." In the case of organizations and from a pragmatic viewpoint, the "anticipation" of the constitutive declarative space and the networks of actual articulations of commitments may be observed; criteria of pragmatic success and failure can be established. If these conditions are satisfied, the analysis of actual interactions can then be carried out.

The professional discipline of systems analysis has focused on routine structured processes. Problems of volume and routine are frequently studied, rather than problems of communication. Although it has limitations, this work is making progress in providing middle management with systems for billing, payroll, accounting, inventory control, etc. Systems analysts classify other problems as nonstructured. This prejudice is derived from certain preconceptions of structure. Contrary to these preconceptions, we assert that the institutional structure of the organization is brought into being by the generation of structured conversations.

We believe that in the future *commitment analysis* will become a leading theory. It will simultaneously shape the design of organizations and systems. Of course, we are not asserting that these pragmatic criteria will be the only criteria for organizational design. There are many others, including the importance of power as a motivating criterion. In any case, effective performance in the production of goods and services will always be one of the essential criteria for analysis and design.

SOME FINAL OBSERVATIONS

It is our belief that the above discussion presents a fundamental and pervasive theory of the way managers and other knowledge workers actually operate in the office environment. This theory is considerably less naïve than the traditional view that managers communicate by transferring information and make decisions by selecting from alternatives. The theory also provides a comprehensive context for considering the application of information technology tools to improve the performance of people in organizational offices. History and tradition suggest that relatively independent developments in data processing, Management Information Systems, word processing, and Decision Support Systems will be followed by a great deal of effort to interface these separate systems. We suggest that it is more productive to start with an integrated theory of the activities of knowledge workers and then design the tools and systems to improve their performance.

However attractive that sounds, we wish to conclude our paper with some warnings about the limitations of the present approach. It seems important to anticipate and correct misunderstandings

that could arise in the context of this paper. We have talked about the importance, pitfalls, and problems of human communication through the use of computer networks, and we have dealt with some problems of design. But we want to reiterate the importance of face to face conversations, or at least oral conversations over distances. We believe that we require these kinds of interactions in order to remain human beings endowed with a rich variety of moods and a broad understanding. A nefarious danger would be to augment computer interactions without also expanding richer forms of direct contact. We may discover that idle talk is perhaps an important human need. At least we want to see this claim in print.

The role of texts in organizations cannot be reduced to the notion of reports as accounts of past acts. There are all kinds of reports and analyses about different issues, markets, etc. Reports are produced through different networks of consensus, including those that are neither necessarily specific nor required by the organization (such as general consultant newsletters and magazine or journal articles). In spite of this, such reports are solicited and resources are expended in their production and gathering. In our opinion they constitute an important part of the process of permanent interpretation of the possibilities of organizations although the modern business culture misrepresents such reports as a certain kind of poor scientific prediction. One of the fruits of the notion of conversation and understanding will be to open up a space for a more serious examination of these interpretative practices.

Our final remark deals with the forms of discussion and participation that take place in an organization. The idea of decision making being like choosing, the notion of a hierarchy of managers as gatherers of information, and the exclusion of the lower members of a hierarchy from the process of discussion have brought forth many problems for the attitudes of every member of an organization. It is apparent that the Japanese mind works in a different way. Discussion is not necessarily an attack of the capacity for resolution and authority. Its introduction in an organization does not diminish the opportunity and velocity of decision making. It may enrich the quality of the motivation of workers, hindering future break-downs generated by unsatisfactory conversations. This consideration points to the basic process of the formation of collective understanding. We are not arguing here from the perspective of ethical or political ideals. These are part of a broader discussion, outside our present realm of concerns.

REFERENCES

Austin, J.L. (1978) How to Do Things with Words. 2nd Edition. Cambridge, Massachusetts: Harvard University Press.

Bair, J.H. (1978) Communication in the Office-of-the-Future: Where the Real Payoff May Be. Paper presented at the Fourth International Computer Communications Conference. Kyoto, Japan. August 1978.

Flores, C.F. Management and Communication. In preparation.

Heidegger, M. (1962) Being and Time. New York: Harper and Row.

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

Morgan, H.L., and J.U. Soden (1973) Understanding MIS Failures. In Proceedings of the Wharton Conference on Research on Computers in Organizations. Data Base (ACM) 5(2-4): 157-171.

Rogers, E.M., and R.A. Rogers (1976) Communication in Organizations. New York: The Free Press.

Searle, J.R. (1969) Speech Acts: An Essay in the Philosophy of Language. London: Cambridge University Press.

A LOOK BACK AT AN OFFICE OF THE FUTURE

Les Earnest
Stanford University

The letters of invitation to this conference opened with the statement, "The area of Decision Support Systems (DSS) focuses on rather small interactive computer systems ...." As an unrepentant "big computer" person, I feel I must register some protest against this formulation. Although we use small computers in my laboratory and their role is growing, I am convinced that big machines will continue to play a major role in the development of Decision Support Systems. Fortunately, it is not necessary to squabble over the "small machine" versus "shared big machine" issue if we focus instead on what is to be accomplished. As has been noted in other contexts, "It isn't the size that matters, it's what you do with it!"

While I use computers every day and occasionally develop special programs and data base systems in support of my management tasks, I rely most often on computer text files to deal with unstructured situations. I shall focus here on the use of such files to support analysis, planning, and decision making. Specifically, I shall examine techniques currently used for text preparation and printing, news services, network communications, and electronic journalism.

SAIL RESEARCH AND DEVELOPMENT

The Stanford Artificial Intelligence Laboratory (SAIL) is a major element of the Computer Science Department at Stanford. Since it was formed in 1965, the principal research and development interests of SAIL have been in the following areas:

— formal reasoning (representation theory and "common sense" reasoning)
— heuristic programming (symbolic computation, theorem proving)

— image understanding (3D perception)
— robotics (manipulator control, visual navigation)
— speech understanding
— natural language (text) understanding, including machine translation
— chemical inference (interpreting mass spectrograms)
— mathematical theory of computation
— program verification
— computer music (quadraphonic synthesis)

In support of these activities, a number of other projects have been undertaken. Some have developed goals of their own, such as

— design of audio-visual display systems for computer interaction
— development of text and graphical editors and document compilers
— design of robotic equipment (manipulators, remote-controlled vehicle)
— computer-aided design systems for digital logic
— design of advanced computer systems
— programming language design and compiler development

Some of these projects have led to commercial products, such as the DECsystem 20 computers, the electromechanical arms manufactured by Unimation, and a new kind of electronic music synthesizer being produced by Yahama. More technological spinoffs appear imminent.

INTERACTIVE COMPUTER FACILITIES

A sizable computer facility has been developed at SAIL. It currently uses DECsystem 10 and 20 computers, two VAX computers, and an assortment of mini- and micro-computers. From the beginning, the facility was designed to support full (upper/lower case) character sets on both terminals and the line printer. Since 1971 there have been display terminals in every office, with some novel features. The displays can show not only text and graphics (within the available resolution of 480 × 512 pixels), but television as well—including commercial television broadcasts, televised classes and seminars originating on the Stanford campus, images originating from television cameras within the laboratory (e.g., from robotics experiments) and synthetic images from the computer.

A computer-controlled video switch permits users to select whatever sources they wish to see from among 39 channels, at any terminal in the building. This allows individuals to continue interacting with the computer as they wander through the offices, or to share images with someone in another part of the building in conjunction with a telephone discussion. An audio switch and loudspeaker permit selection of any of sixteen audio sources at each terminal, including computer-synthesized sounds, public address announcements, television audio, radio news broadcasts, various kinds of music, or silence.

In addition to the office terminals, about one-third of the staff members have display terminals at home. These text-only terminals connect to the computer via switched telephone circuits using split-speed modems (1200/150 baud). The combination of office and home terminals makes it possible for staff members to "stay connected" to the computer almost as much as they wish. We find that this greatly enhances the productivity of some individuals and enables them to stay in touch with each other via computer "mail" even though they may be working on quite different schedules.

TEXT PREPARATION

The bulk of all information that is entered into the computer originates as text files. This is true whether the material being entered is a document, a computer program, or a data base. Nearly all documentation originating in our laboratory is entered directly into the computer by the authors, eliminating the need for stenographers. It has not been necessary to coerce people to do their own typing—they are seduced by the combination of display terminals in every office, a display-oriented text editor, and a nearby printer that produces good quality output in whatever type styles the author specifies. Even people who have not learned to type before find it expedient to do so.

I am convinced that this "do-it-yourself" style of interaction should and will become standard in most offices of the future. For anything but very routine correspondence, the ability to directly control and rearrange word and paragraph placement, as permitted by a good text editing program, makes this mode of text entry far more efficient than the traditional "dictate-type-edit-retype-..." or "write-type-edit-retype-..." cycles.

What about a person who depends on a secretary to correct spelling errors? A computer program that checks spelling generally does a better job. We have been using such programs since 1969 and find them to be helpful adjuncts to proofreading. Of course, secretaries are still needed in an "on-line" office, though not as many as in a conventional office.* They must be able to type, and must learn to use the computer terminal. Training new secretaries and others in the use of the text editor has been surprisingly easy. Our primary text editor, succinctly named "E," has a tutorial mode that leads the user through the basic editing functions. Very few people take more than an hour to learn enough to begin using the computer productively.

Getting everyone to use the computer has a very beneficial side effect: noisy typewriters go unused most of the time. They

---

*The laboratory has a staff of slightly over 100 people (faculty, professional staff, and Ph.D. students), but only three secretaries. This unusual ratio of professional to support staff is made practical by using the computer to provide many support functions.

are needed only to fill out certain forms that are used by the University and the government. We look forward to the time when these agencies will be connected to computer networks so that we can get rid of typewriters altogether.

One text editor that is available to us shows text on the screen in the same form as it will appear on paper, including variable-width characters, italics, and various type styles. While this "what you see is what you get" style of editing has certain advantages for composing letters and other simple documents, we assemble most large documents as "manuscript" files, which are then compiled into a printed version by one of the available document compilers.

Looking to the future, a widely recognized goal is to develop an "automatic secretary"—a speech recognition system that translates spoken words into text files. Unfortunately, existing systems work only with vocabularies of at most a few hundred words and are rather slow and unreliable at that. Keyboards will remain the most efficient way to enter text for the foreseeable future.

DOCUMENT COMPILERS

The primary task of a document compiler is to do formatting, i.e., to fit the text into the available space on the printed page. The document compilers used at SAIL also provide certain clerical services such as the numbering of pages, sections, and figures. Some can also automatically compile a table of contents or subject index and can determine appropriate values for cross references at the time of compilation.

"Pub," our first document compiler, was developed in the early 1970s. It has a number of quirks, but is extremely powerful—it includes most of Algol 60 as a subset of its commands. Unfortunately, secretaries and other non-programmers find it difficult to understand. Other compilers called "Pox" and "Scribe" have substantial numbers of adherents. Don Knuth's new compiler, called *Tex*, has become an instant success, particularly among people doing typesetting of mathematical expressions. Unfortunately, no single document compiler dominates in all respects. Since this is a relatively new field, it will take time for systems to be developed that are both complete and balanced.

Bringing all this technology to bear on preparing documents does not necessarily speed up the process; since it is relatively easy to modify a document and produce a new version, authors usually spend more time polishing the text than if the material had been typed on paper in the first place. As a consequence, documents produced in this way tend to be better written.

PRINTING

There are several printer-plotters connected to the SAIL computer that are capable of printing text, line drawings, and half-tone images in a variety of formats (21.6 cm to 56 cm in width). A Xerox Graphics Printer is most heavily used; this is a high-speed facsimile printer that has been adapted for use as a printer. There is also a slow but very high resolution photographic printer (Alphatype) connected to the computer for use in quality printing, such as books. There are no "built-in" character sets in these devices—all graphical constructs, including character sets, are done in software. Consequently, text can be printed in a variety of sizes and styles. Indeed, there are over 300 fonts available currently and the number increases weekly.

In addition to the latin alphabet (in many styles), there are a number of other symbol sets available, including exotic mathematical symbols, Greek, Cyrillic, and special symbols for most other European languages—even Old Icelandic. Hebrew and Arabic are available, as is a subset of Chinese and Mayan. There are also some whimsical character sets, such as the Tengwar, used in Tolkien's novels (*Lord of the Rings*). Scholars of old languages particularly appreciate the ability to print source materials in the original character set. Mathematicians enjoy the ease with which new symbols can be invented to embellish their art.

NEWS SERVICE

Since 1972, the SAIL computer has offered access to newswire information in several modes. The system allows users to retrieve stories up to two weeks old on the basis of an automatic keyword indexing system. The indexing system has the unique feature of permitting stories to be retrieved by any keywords at the moment the stories appear.

Two newswires (Associated Press and New York Times) are connected to the computer. Users may link their displays to one or both of them, to view the latest news as it arrives. This mode of news access is useful when a major story is unfolding. Most of the time, however, the stories appearing on the newswires at a given moment are not very interesting, so this way of accessing the news is of limited value.

As news stories arrive, special programs store them on disk files and exhaustively index the text. That is, essentially all words are treated as potential keywords. The programs have a list of common words (such as "a", "an", "the", etc.) that are ignored for indexing purposes. The indexes are stored in disk files. Users may specify any Boolean combination of keywords. The usual way of querying is to specify just a single word—such as "Helens"—to find out what the volcano is doing. The retrieval program immediately shows how many stories in the prescribed time period use the specified keyword(s). If too many stories are

found, the user may apply additional constraints (e.g., "explosion") to reduce the number of stories retrieved.

In addition to this *ad hoc* style of news access, users may leave standing requests, so that whenever a story arrives that contains a certain keyword or combination of keywords, they will be notified immediately by electronic mail. This automatic "clipping service" is particularly useful if one is trying to monitor happenings of a certain kind, such as nuclear accidents, whenever and wherever they occur.

The news service has been running on the SAIL computer for eight years with no human assistance. In contrast, all commercial news retrieval services with which I am familiar require a sizable number of people to abstract or index the material to be retrieved.

A "front page" is one feature that this computerized news service does not provide—that is, a guide to the current most important news *from some viewpoint*. This requires human editing at present, but would not be difficult to do. In fact, since a "front page" is simply a list of stories, it should be feasible to offer several front pages to the readers—e.g., a conservative front page, a leftist front page, a sportsman's front page, etc.

Another service that would be interesting to have in future commercial versions of this system is "instant letters to the editor." Given that readers will be accessing the news service through a display terminal, it would be relatively easy to permit them, in effect, to place commentary in footnotes to the text. There would have to be a small fee for storing such comments. Other users could then read the text either with or without footnotes, as they choose, or perhaps with selected footnotes. For example, if a story mentions a certain person by name, it would be possible to learn whether he had filed a remark. Or one could ask for any statements filed by government leaders in a certain class. Overall, the instant feedback that will be possible should greatly enhance discussions of controversial topics. I look forward to the widespread use of electronic news services of this type, in the hands of an informed populace.

Other elements of the newspaper also lend themselves to electronic treatment. For example, classified advertisements can be handled quite efficiently by computer inquiry systems. More complete descriptions of the things for sale can be given without incurring much expense—making it unnecessary to waste so much paper for printing ads. The same query techniques that are outlined above for the news service should work rather well for classified ads. For example, if you are interested in buying a certain rare kind of automobile, you will be able to leave a standing request so that whenever one appears in the listing, you will be notified immediately.

NETWORKING: ARPANET AND DIALNET

The SAIL computer has been connected to the Arpanet since 1971. This network now connects about 200 computers located

throughout the USA, with a few more in Hawaii and Europe connected by satellite links. It has been extremely useful to our staff in a number of ways. The most heavily used feature is electronic mail, which permits individuals at great distances from one another to collaborate in writing articles, using minute-by-minute interactions when needed.

The ability to share specialized resources is also quite valuable. For example, there are some computers at MIT with programs specialized for symbolic computation, called "Macsyma". Instead of having to replicate that capability on each computer where it is needed, users can remotely access the Macsyma machine via the Arpanet and may use it as if it were part of their own computer.

While Arpanet provides extremely valuable services, it poses some economic and administrative problems. The cost of providing Arpanet service to a given computer is on the order of $100,000 per year, so it is not sensible to connect systems unless they will use the network rather heavily. An alternative networking scheme called "Dialnet" appears attractive for low volume users. The basic idea is to use conventional switched telephone lines with moderate data rate modems (1200 to 4800 baud) to connect computers as needed. This system has been under development at SAIL since July 1977, under the sponsorship of the National Science Foundation. It now runs on three computers at Stanford. A set of protocols has been developed that enables a computer user to send messages to users of other computers to transmit files between his own and other computers, and to use other time-shared computers directly—all using the existing dial-up telephone network. No formal network administration is required. The users of any computer implementing the protocols can communicate with the users of any other computer implementing them—anywhere in the world.

The hardware cost is typically about $2000, depending on the type of system and how difficult it is to connect devices to the computer. For time-sharing systems, an automatic telephone dialer allows the system to initiate calls. For small single-user systems where economy is paramount, the user can do his own dialing. The only disadvantage compared with the Arpanet is generally lower speed.

ELECTRONIC JOURNALISM

Users of computer communication networks such as Arpanet are beginning to interact in new ways that appear to be a preview of the "electronic journalism" of the future. Given such a network, it is relatively easy to construct distribution lists of people that share some common interest and to begin a "multilog" (a many-way conversation). That is exactly what is happening—people at various institutions around the world who happen to share an interest in some technical area, or even some recreational activity, are locating each other on the network and communicating regularly.

The simplest form of association of this sort consists of a distribution list that someone maintains. Each person who has something to say sends a message to the list and it is distributed almost instantly.

A slightly more advanced form involves an editor. All messages to the group are directed to the editor. At the end of each day, he assembles them in some kind of logical order, edits or eliminates some of the material, and turns it over to a program that mails it to everyone on the list in the middle of the night. Thus all subscribers get a new edition in the morning that was written the day before.

My name was put on one of these lists, the Human-nets list, quite by accident. I have since become addicted to reading it each morning. The Human-nets group is discussing technological, social, and political problems associated with future worldwide communication networks. There are usually several topics under discussion at any given time and some discussions go on for weeks or recur. Of course, the skill of the editor has much to do with the quality of the journal.

One recent whimsical but stimulating discussion centered on "games one million people can play." The idea was to develop interactive games in which people participate either as individuals or as members of a team. The game is played via a communication network, so that participants may be widely separated. An advanced version of the "Dungeons and Dragons" game was thought to be a good candidate to stimulate this kind of participation.

In a more practical vein, public discussion of social issues via electronic journalism appears to offer some advantages over traditional publication forms, particularly for issues that are transient in nature.

The quickness of communication carries with it a problem, however; we have discovered that it is much easier for people to lose their tempers in this new form of journalism than in slower systems. The problem clearly arises from perceiving mild disagreements as insults when they are quickly distributed to hundreds of people. This phenomenon has been observed enough times, even involving normally cool-headed people, that it has been given a label: "flaming." The Human-nets editor has learned to suppress material that seems to be "flaming."

CONCLUSION

The effects on a research group of having highly interactive computer support have been reviewed. Decision makers have found it convenient to use the system personally, both as a communication device and for analytical support. While small computers can provide some support capabilities without having an external communication capability, I believe that the primary value of such devices will be to provide a "porthole on the world." The development of this capability should be given priority in the design of future systems.

INSTALLING A DECISION SUPPORT SYSTEM: IMPLICATIONS FOR RESEARCH*

Ephraim R. McLean
University of California

Thomas F. Riesing
Nolan, Norton and Company

INTRODUCTION

Managers are continually searching for ways to become more effective. Increasing attention is being given to decision support systems (DSS), as distinct from more traditional transaction-based data processing systems. At the same time, the costs of DSS technologies, which might lead to improved effectiveness, are rapidly decreasing. The problem is to harness DSS technologies in a way that produces the desired improvement in the performance of managers.

As is often the case, decision support systems are not being developed within the quiet confines of the university, but rather in the fast-paced world of industry. Companies are undertaking the development of DSS, not in a quest for knowledge, but because these new systems are needed. DSS are real systems, not toy ones.

Because of the unique nature of each company's DSS activity, it is often difficult to make generalizations. Research is sorely needed which will help future DSS developers avoid the mistakes of others and insure maximum benefit for the effort expended. One way to begin is to study existing decision support systems and to identify key research questions. This paper describes one such effort, focusing on Citibank's MAPP (Managerial Analysis

---

*This paper is based upon research conducted at Citibank (North America) during the summer of 1976. Some research results were presented at the Conference on Decision Support Systems, San Jose, California, January 24-26, 1977, and subsequently published in Database 8(3): 9-14. The authors wish to express their thanks to the officers of the bank, especially Martin Foont, Assistant Vice-President, whose support made the study possible and whose openness and candor allowed its publication.

for Profit Planning) system; eight research hypotheses are derived from that experience.

A DESCRIPTION OF THE MAPP SYSTEM

The MAPP system was developed and installed at Citibank (formerly First National City Bank) in New York City in mid-1975. It was designed to support financial planning and budgeting throughout the bank and replaced an earlier manual system that had been in use for some time. MAPP was used during the 1975-1976 period and then discontinued. The reasons for this action were varied, some reflecting major organizational changes within the bank and others resulting from characteristics of the MAPP system itself. Both factors will be discussed later.

Citibank reviewed its budget preparation procedures in 1974 and realized that many of its budgeting problems were common to other types of organizations. It was clear that if a successful solution could be found within the bank, the same approach might be of interest to others. A natural market for such a system might be other banks, brokerage houses, insurance companies, and various financial institutions. In addition to financially oriented organizations, it was also possible that manufacturing and service organizations could make use of a system like MAPP—provided that the fundamental design was sufficiently flexible and general. The more the information structure of MAPP was tailored to the Citibank environment, the more difficult it would be for other organizations to use the system. However, if it were designed without a built-in organizational, product, or financial structure, then users could adapt the system to meet their organizational needs merely by tailoring the input.

One of the key decisions in the development of MAPP was to let the user rather than the designer control the information structure. This philosophy greatly expanded the system's potential usefulness and scope of application. However, this benefit was not without costs. If the system had been made Citibank-specific, a number of problems might have been avoided. The design would have been simpler and less demanding for users. As it was, certain trade-offs and compromises still had to be made in order to insure successful implementation at the bank within the time horizon available.

System Features

In simplest terms, the MAPP system replaced the cost accounting function within the bank. It established a procedure for defining products, identified the costs of producing these products, allowed analysts to determine how resources might be shifted among products, and, finally, helped the departments that produced the products to prepare budgets. This spectrum of activities, ranging from structured to unstructured, must be performed in almost every type of organization. Some steps, however, are frequently carried out only implicitly. Underlying

assumptions are never brought to light and challenged; alternative courses of action are not examined and explicitly considered.

A lack of detailed information is usually not a handicap for managers who operate in a fairly stable environment and who have extensive experience with the products or services being offered. They can make decisions about pricing, levels of marketing effort, budgetary objectives, and other items solely on the basis of their intuition and judgment.

Until recently, this was particularly true for managers in the banking and brokerage industries. They paid more attention to the customer "relationship" than to the costs of running the "back office" and to the definition and detailed costing of specific products. However, financial institutions are now beginning to look for ways to measure and control their costs. This concern for the cost accounting function, while relatively recent for banks, has a much longer history in other industries. Both manual and mechanized systems have been devised in an attempt to furnish management with better information for financial planning and control. The MAPP system was developed to provide such a service at Citibank. It was hoped that MAPP would assist in product definition, cost assignment and organizational definition, financial planning, and budget preparation. Each of these areas of application will be discussed in more detail below.

Product Definition. To those in manufacturing, the concept of a "product" is so well established that the need to define products appears superfluous. Because they are physical, tangible entities, products are reasonably well understood throughout the organization. Of course, the production function may think in terms of fenders and steering wheels, while the sales function may think of station wagons and sedans. Nevertheless, the ability to trace the flow of products--and, more important for our purposes here, to assign costs to these products--is sufficiently well developed to permit straightforward product-based cost control.

In service industries, "products"—e.g., arranging for a loan, repairing a television set, or giving an inoculation—are not well defined. The costs associated with such service "products" are similarly difficult to identify and monitor. However, without a detailed product structure, it is extremely difficult to perform cost control and product profitability reporting.

The designers of MAPP recognized that the first step in the development of a financial planning and budget system must be definition of an organization's products. In the case of Citibank, this meant that over 500 services—or "products"—had to be defined. For some bank managers, this proved to be an enlightening exercise in itself. For instance, it became clear that "checking accounts" are not a single product but a group of products, each with distinctive characteristics and costs; conversely, excessive differentiation may mask certain products that are virtually identical. Thus, the first feature MAPP provided was a systematic way for managers to define the products offered by the bank.

Cost Assignment and Organizational Definition. Once the product structure had been established, the next step was to identify the costs, both fixed and variable, direct and indirect, associated with the products. The process of assigning costs was particularly critical, for all subsequent decisions would be based upon them; confidence in the system as a whole suffered when cost assignment was done in an offhand manner.

Before the development of MAPP, the cost allocation function was carried out by a staff of cost accountants or financial analysts at Citibank. This approach has advantages and disadvantages. Cost accountants, both by training and by inclination, are well suited to "number massaging." Their results are likely to be objective, consistent, and reasonably accurate. However, as staff specialists, accountants do not have the intimate knowledge of the products and the organization that line managers possess. Line managers are in a far better position to judge which costs should be apportioned to which products or departments. This is especially true in service industries, where "product" definition is likely to be matter of judgment in the first place. For these reasons MAPP was designed to take the cost allocation process out of the hands of accountants and to give it instead to managers of financial reporting centers (FRCS, as MAPP calls them) within the organization.

MAPP also had to capture the organizational structure in order to facilitate cost and budget consolidations. Like the product structure, the organizational tree structure was designed to be completely user-generated. Reporting relationships, departmental structures, and the relationship between products and departments were established by designated controllers within each group or division. These controllers also exercised a measure of discipline over the system: by the very nature of cost allocation, there must be a "sending" department and a "receiving" department; the controllers made sure that transfers were mutually agreeable, and in cases of disagreement they arranged for face-to-face mediation between the managers involved.

Financial Planning. An organization with limited financial resources must decide how to allocate these resources. Ideally, decisions about allocation should result from an iterative planning process. This could take the following form. In the "top-down" phase, top management sets institutional performance objectives and targets for profitability and return on investment. These are then conveyed down through the organization, and refined at each lower level. In the "bottom-up" phase, operating managers project profit plans for their areas of responsibility for the upcoming period, usually a year, using information on costs, product mixes, prices, revenues, and other factors. Typically a gap exists between the two sets of estimates and managers must explore ways of closing it. Alternative plans must be considered and the effects of shifts in volumes, prices, service levels, product mixes, and other factors must be examined.

In many organizations this is usually done in the following way. Recognizing that budget setting is largely a political

process, managers argue for all they can get; when forced to accept a lower figure, they then shift their attention to cutting costs in whatever way possible to "make budget."

MAPP was designed to help both in projecting expenditures and in meeting budget restrictions. Prior to the finalization of the budget, MAPP provided information that allowed managers to explore the impact of different planning assumptions. In this way, they could "home in" on those aspects of their operation that were most susceptible to change and improvement. After the budget had been set, the same techniques could be employed to investigate new opportunities or to cope with changed circumstances.

Of course, such evaluations depend on the quality of the data base. If data are carelessly assembled, the potential usefulness of the analyses is substantially diminished. As is true of many such systems, the quality of the data that the system contains and the use that is made of the data are directly related to the commitment, competence, and enthusiasm of the user.

Budget Preparation. A budget is a projection of expenditures for some future period; it is a commitment on the part of management that, given certain assumptions, spending will stay within certain limits. If all steps have been performed properly and all data correctly entered, the final budget becomes a natural by-product of the financial planning cycle described above. Even managers who perceived the total size of their budget to be fixed use MAPP to make trade-offs within cost categories and to explore ways to meet the mandated budget.

In most organizations the planning and budgeting process is a time-consuming, tedious task. Frequently, the cycle must be repeated two or three times, and the sheer effort of manually consolidating the various individual budgets can take several man-months. At Citibank, this was not only tedious work, but also highly prone to error; the staff was usually so glad to have completed the process once, that it had little desire to go back and make any changes. With MAPP, extensions and consolidations became trivial to perform. Although the justification for using MAPP rested on the improvements it could make in managerial effectiveness, many felt that it could have been justified solely on the basis of reductions in the accounting staffs needed to perform budget consolidations.

THE DEVELOPMENT OF MAPP

In January of 1975, the development of a system like MAPP was proposed to the bank's top management. The system's potential for becoming a marketable product was stressed as much as its value for internal use. No strict dollar benefits were claimed for the system.

The proposal received the enthusiastic endorsement of the bank's executive vice-president, who also served as the head of

the Operating Group. In retrospect, one may question how widely this commitment was shared by other officers in the Group. In any event, approval was given, with the understanding that the system would be ready for use by the start of the bank's annual planning cycle in July.

In developing new systems, Citibank has a long-standing policy of using the services of outside consultants and programming specialists. In this way the bank circumvents the need to maintain large permanent staffs. The policy also permits the bank to be flexible in taking advantage of new technologies, programming techniques, and so forth. Several software firms and time-sharing companies were contacted to program the MAPP system. The companies were free to choose their programming language, and could propose using their own computers or Citibank computers to run the program. Thus bids could include both the costs of software development and computer time, or the costs of software development alone. Use of one's own computer is more attractive to most vendors in the long run; it is possible that some companies made their programming bids artificially low in hopes of generating future operating revenues.

In any event, a bid of $300,000 was received for a COBOL-based system, a bid of nearly $200,000 for a FORTRAN-based system, and a bid of slightly over $20,000 for an APL-based system! Even if price were not sufficient reason to choose APL as the development language, another important reason came to light: none of the non-APL bidders would even consider the specified time frame. They wanted at least a year, and yet only three months were available. Thus only APL-bidders were considered, and the final decision was made on the basis of price alone. The winning bidder committed three full-time programmers to the task of software development, with the understanding that the resultant system would be run on the company's own computer. It was agreed that any subsequent commercial marketing of the system would be a joint venture.

As the development of the MAPP system began, it became apparent that initially the immediate internal needs of Citibank would dominate the project. The system had to be operational by July 1, and nothing was permitted to interfere with this goal. The project was largely in the hands of the Director of the bank's Decision Support Systems (DSS) Department. He was responsible for assuring that the outside vendor performed according to the contract and that the internal installation proceeded as planned. With less than 90 days available, it was clear that equal attention could not be given to all phases, i.e., that some trade-offs had to be made. Unfortunately, these compromises produced a whole cluster of problems. These are reviewed briefly in the following sections.

Design

The first problem concerned the design phase. Although the overall concept of the system was well understood, many details

of the design had not been specified. Because of the power of APL and the confidence of the APL programmers, coding began almost immediately, without a detailed system design. As a result, individual programmers made many decisions about program and file structures without considering how their decisions might affect other parts of the system. Choices were usually made for the sake of expediency—"to get the job done." Some decisions, especially those affecting the organization and contents of files, were fairly fundamental; in light of subsequent design efforts, they proved to be less than ideal.

The multilevel prompts displayed to the user during a session on the terminal presented another difficulty. MAPP provided for three levels of prompts: a normal level (the default), an abbreviated level for experienced users, and a detailed level for novice users. The users could select the level that they wished to see displayed. The designers felt that the three levels of prompts would virtually eliminate the need for reference manuals, training sessions, and paper documentation. Unfortunately, this did not prove to be the case; without manuals and briefing sessions (which were not provided until the time of the 1976 planning cycle), users did not understand MAPP's basic purpose and did not know how to conduct a simple session at the terminal. The *mechanics* of the system were explained by the prompts; the *reasons* for the system were not.

In addition to this strategic shortcoming, the prompt system suffered from the same lack of an overall design that affected the file structures. Ideally the prompts should have been created in a separate file, where they could have been readily reviewed, modified, and extended as needed. Instead they were embedded in the programs in which they were used, making modifications difficult and prone to error. It also became apparent that programmers in general (and probably APL programmers in particular) are not noted for their literary skills. Writing prompts that are clear, concise, and "user-friendly" is a special talent; to compound the problem, the MAPP programmers were required to create not one, but three, levels of prompts. Given the time pressures, it is not surprising to find that one prompt was often used for all three levels.

Another problem with design concerned the location of the project management. For the first month of the project, the vendor's APL programmers were off-site, and supervision was carried out by the vendor. Without close daily contact, the bank found it extremely difficult to monitor progress. Because a detailed design was lacking, it was almost impossible to establish project milestones. This situation was corrected when the bank insisted that the contract programmers be relocated to the bank's premises. After this move, the bank's Director of DSS became the *de facto* programming leader in addition to his other duties.

Implementation

The implementation phase of the project posed even more serious problems. In spite of the commitment of the head of the

Operating Group, there was little time, and even less manpower, to introduce the system properly to rank-and-file bank managers. Most of the DSS Director's energies were spent in making sure that the users had something to use! For the reasons indicated above, the training sessions were minimal; user documentation, except for that which was built into the system, was equally sparse. As a result, users' perceptions of the purpose and rationale of the system varied widely. Some saw it merely as a more efficient way to perform the odious annual task of cost allocation. Others realized that if MAPP produced reliable information, it could aid in managerial decision making. But many bank managers used MAPP simply because they had no other choice; the centralized staff group that had formerly performed cost allocations was disbanded shortly before the MAPP project began. Thus managers often produced data for MAPP about their products and processes with little understanding of or interest in the task. About 35 of the 170 financial reporting centers had to reenter their cost allocations to correct gross errors made in the first round.

In spite of all these problems, the system *was* operational by July 1; if any slippage had occurred in meeting this deadline, a whole year might have been lost. However, there was little time to test the system before it went into production and users encountered numerous bugs. Although this was frustrating and tended to erode users' confidence in the system, most bugs were corrected almost immediately, in large measure because of the flexibility and power of APL and the skill of the APL programmers.

In order to compensate for the lack of training sessions and formal documentation, a MAPP hot-line was established. Users experiencing problems with the system could telephone an APL programmer during business hours; the programmer would review the application in question step by step with the user, to determine the exact circumstances that caused the problem. If the problem stemmed from the user's lack of knowledge, the programmer would explain the correct procedure—at the same time making notes for use in preparing future reference materials. If the problem stemmed from a bug in the program(s), the programmer would sign onto the system and fix the bug "on the fly." It is doubtful whether it would have been possible to correct system problems in "real time" if a language other than APL had been used. (In fact, as was pointed out earlier, the system probably would not have been ready for use by the July deadline if the APL language had not been chosen.)

Session logs were maintained as a by-product of the MAPP terminal prompting monitor. Thus the sequence of inputs made by users during each terminal session was readily available for review. The session logs served two purposes. First, if a user reported a problem after it happened, the session log could be used to recreate the exact situation that obtained when the error occurred. Second, by studying session logs for a number of sessions, bottlenecks and unexpected usage patterns could be identified; this made it possible to recode portions of the system for greater efficiency.

After the annual planning cycle was completed in the fall, the project team made permanent changes in the programs and implemented features that had previously been deferred. It was at this time that earlier design shortcuts became obvious. But rather than stepping back and taking a new look at the overall design, the team worked very hard to make the existing design work. In retrospect, this was a costly decision. The team would have been better advised to "start from scratch" rather than to attempt a massive rework. More than twice the manpower was spent in attempting to modify the system for the 1976 summer planning cycle than had been needed to create the system in the first place.

USERS' REACTIONS TO MAPP

In order to obtain an independent perspective on the MAPP system, E.R. McLean was retained to conduct a study within the bank in the summer of 1976. Twenty-two individuals from both the Operating Group (the "back office") and the Banking Groups (the "front office") were interviewed with the aid of a structured questionnaire. The titles of the persons interviewed ranged from "manager" to "senior vice-president." Some had little or no direct experience with the MAPP system, but all had been involved in the 1975 cost allocation and budgeting cycle. Their reactions to MAPP provide valuable insights about the general nature of decision support systems.

In spite of the problems described above, there was a general consensus that the use of MAPP to perform cost allocations was an improvement over the previous manual method. The disciplined approach built into the MAPP system assured more complete and accurate allocations. MAPP also brought to light cases of gross misallocation in the past; one such instance involved the allocation of EDP costs. The true costs of providing certain computer services were found to be much higher than had been realized previously. The bank officer who had been receiving EDP charges subsequently used the MAPP figures to justify his request for a computer with which to allocate EDP costs on a decentralized and more cost effective basis.

Some users reported that MAPP had helped them to develop a better sense of the nature of a "product" within the banking environment; others mentioned that the system helped to promote communication between the operations and banking groups. One manager found that certain branch activities seemed out of line and he initiated a study to determine the cause. "Without MAPP," he stated, "the need for this study might have taken a very long time to come to light." Another manager, forced by MAPP to examine his product and subproduct structure, realized that many products were not unique. As a result, he was able to reclassify 2,060 different billing items into only 30 items—certainly a much more manageable number.

Aside from these few exceptions, most managers could not point to any *management decisions* that had been influenced by

MAPP. They felt it was a fine system—for someone else! As reasons for this finding were explored, four points emerged—related to fixed budgets, reorganization, product pricing, and system complexity.

Fixed Budgets

Unlike many other organizations, an exclusively top-down process was used for budget setting within the Operating Group at Citibank. The expression "flat-to-prior-year" was used to describe how budgets were set. In other words, a manager was expected to expend no more in the current period than he had spent in the previous period, even if volumes or product mixes changed. Because outcomes were predetermined, managers had very little interest in exploring budget alternatives or playing "what if?" games.

Reorganization

A second reason why MAPP was not used in management decision making concerned cost allocation. Most managers agreed that it was worthwhile to identify the costs incurred in producing a product or servicing a customer. It was also clear that if most costs accrued in a large central part of the organization, then some allocation scheme was needed. However, shortly after the decision to develop MAPP took place, a major effort was made to decentralize the bank. This reorganization, which was given the code name of Project Paradise*, had a profound impact. The monolithic 8,000-person Operating Group was subdivided into more manageable units, each unit linked directly to the banking unit it served. This decentralization greatly diminished the need for detailed allocation. Operations and marketing groups were clustered around well-defined customer or product groups; thus many of the benefits that the MAPP system might have yielded were achieved in other ways.

Product Pricing

Pricing mechanisms acted as a third deterrent to the full implementation of MAPP as a decision aid. Because MAPP's features allowed marketing managers to get a good idea of the true costs of the bank's products, it was hoped that MAPP would enable managers to set prices more intelligently. However, some managers pointed out during their interviews that

— most prices were market-determined and not cost-determined;
— in the case of some products, customers were not very sensitive to individual prices, for the "total relationship" was more important;

---

*This code name gave rise to such questions as "Are you in Paradise (i.e., decentralized) yet?"

— the margins on some products were so large that a wide error could be tolerated;
— in cases in which products were "paid for" with compensating balances, balances required for credit purposes sometimes so far exceeded those needed to pay for the services that the issue became moot.

System Complexity

Finally, the appropriateness of the MAPP technique must be examined. Even managers who were concerned about budget setting, cost allocations, and/or pricing decisions raised questions about the need for a system as complex and sophisticated as MAPP. The feeling was that the added benefits of MAPP were not worth the extra effort. When asked to give his opinion of MAPP, one user replied, "Great; I'm glad the bank developed it." When pressed for specifics, he explained, "MAPP is so complicated and so involved and the data entry is so time-consuming that if we ever needed a justification for becoming 'paradised' (i.e., decentralized), the opportunity to get rid of systems like this should be reason enough!"

IMPLICATIONS FOR DSS RESEARCH

Citibank learned a number of valuable lessons from its experience with the MAPP system—lessons that helped the bank to improve its development efforts on subsequent systems. The Citibank case may also be useful to academics and practitioners who are interested in DSS research. But rather than dwell on what was learned at Citibank,* it is perhaps more useful here to point out what is yet to be learned. A number of aspects of the design and installation of the MAPP system suggest fruitful areas for future research. Five general areas of research questions may be identified: the design process, tools for building systems, user interaction, user education and training, and the organizational setting.

The Design Process

Because of a number of well-documented failures of large, transaction-oriented systems, the importance of good front-end design is increasingly recognized. Without a complete, detailed set of specifications that are clearly understood by prospective users, the chances of developing a successful operational system are greatly diminished. "How do you know where you're going if you don't have a detailed road map?" argue advocates of good front-end design. Clearly, certain aspects of the MAPP system suffered because they had not been sufficiently thought through before coding began.

---

*This was discussed at length in a Database article [Database 8(3): 9-14].

A number of tools and techniques are being developed for those who stress the importance of extensive, careful design activity before programming is begun. Structured design, top-down design, structured walkthroughs, and computer-based design aids (e.g., PSL/PSA), for example, have been touted as valuable aids in the design (and subsequent development) process.

The value and comparative advantage of such approaches are worthy of study in themselves; but the question of concern here is their value in DSS development. Do they help insure a higher probability of success? If so, which tools are best? Is thorough front-end design essential before a DSS is begun? Or are efforts to "get everything right at the outset" likely to be counter-productive? These questions suggest our first hypothesis for research.

> Hypothesis 1: *In building decision support systems, the more attention devoted to the initial design phase, the greater the likelihood of success of the resultant system.*

If research proves that Hypothesis 1 is correct, then a second hypothesis should be investigated.

> Hypothesis 2: *Certain design methodologies are superior to others in building decision support systems.*

In contrast to the front-end design approach, another school maintains that decision support systems *cannot* be specified in advance, for they are, by nature, a growing, changing, and evolving type of system; thus attempts to define all the features in advance are contrary to the very essence of DSS and can be counterproductive. Proponents of this approach argue for prototype systems and evolving design. In contrast to the detailed "road map" approach, they would argue for "finding our way as we go; and if we find something interesting, we'll take that route." Thus the emphasis is on flexibility and adaptability, not on extensive front-end design. For a prototype to work, of course, it must be small. (Unlike the MAPP situation, where the first test usage involved planning for and allocating nearly a quarter of a billion dollars, with well over a hundred bank officers in an on-line mode!) A prototype must have a small group of well-defined users and a reasonable-sized data base. Although it may be difficult to meet these conditions, the following hypothesis merits investigation.

> Hypotheses 3: *Prototype designs, which can be modified as needed, offer the greatest likelihood of success in building decision support systems.*

If this proves correct, then a number of other questions come to the fore. One question, dealing with how prototypes should be built, will be discussed in the next section. Other questions worth investigation include, When does a prototype become "operational"? How is the user base expanded? When, and under what circumstances, should recoding for greater efficiency be undertaken?

Tools for Building Decision Support Systems

As indicated above, one school of DSS designers argues for "getting something up and running" as quickly as possible. To do this, interactive systems are necessary. Conversational, interactive systems are as important for DSS developers as for DSS users. Research by Sachman and others has pointed out the benefits to programmers of working in an interactive environment, rather than taking more traditional batch approaches. An unanswered question is, Does it make any difference *which* interactive system is used?

Supporters of the DEC UNIX or IBM CMS operating systems claim special benefits for their systems. Commercial time-sharing systems like NCSS's NOMAND also have supporters. Finally, BASIC and APL are popular general-purpose languages. APL, in particular, has almost developed into a cult. Systems like MAPP have helped to sustain the legend: using APL as its programming language, MAPP took a quarter of the development time and a tenth of the cost that other approaches would have required. In addition, APL allowed MAPP to be modified easily, even "on the fly" if needed. Is APL, then, an invaluable tool in building decision support systems, or are other interactive approaches equally good? This question leads to our fourth hypothesis.

> Hypothesis 4: *Using APL to build decision support systems offers a distinct advantage over other interactive languages or approaches.*

User Interaction

Hypothesis 4 focuses on the designer's interaction with the system in the process of creating a DSS. But what of the user? Although a few "cultists" have argued that APL is *the* DSS language for designers *and* users, most people recognize that the user interface must be far more "user-friendly" than that provided by straight APL.

The research questions here do not relate only to DSS, but also touch upon the larger area known as human factors engineering (in Europe the term "ergonomics" is often used). Human factors engineering is concerned with all issues related to the user's interaction with the system. Screen formats, keyboard layouts, use of menu selection, color, and many other aspects of the interface can have a potential positive or negative effect on users' willingness to use the system. Because decision support systems have a uniquely *discretionary* character, these issues have a special importance for DSS users. People use these systems because they *want* to, not because they *have* to. They must feel that the system provides an advantage over other alternatives (e.g., doing a task manually). Thus it is essential that the system be perceived as powerful and easy to use.

The area of user interaction gives rise to a number of research questions. Several assertions in need of verification are mentioned below:

Hypothesis 5: *An interactive system will be perceived as being more user-friendly if it*
- *provides menu selection for novice or occasional users,*
- *provides command-type instructions for experienced or frequent users,*
- *provides on-line instructions or documentation on an as-needed basis,*
- *uses open-format input as opposed to structured or table-like input,*
- *provides graphical input and/or output,*
- *uses color displays, and*
- *provides special-purpose function keys on the keyboard.*

User Education and Training

The issue of user "education" should be considered in its broadest context, including appreciation of the system's purpose, commitment to its successful use, and by extension, a feeling of ownership. These are generally discussed under the rubric of "user involvement."

In the Citibank case, there was certainly top management commitment (but not involvement) to MAPP's development and use. At the middle and lower managerial levels both commitment and involvement were lacking. The managers were neither resistant nor hostile; they simply did not know what was being attempted. The cost allocation aspects of MAPP were straightforward enough (even somewhat tedious). But most of the decision support aspects were understood by only a handful of bank officers—hardly enough to sustain its continued use.

In the Citibank situation, these problems could have been rectified by improved user education and training, assuming that the system itself was sound. But what of decision support systems in general? Do they require more (or less) user involvement and education than other types of systems? And what form should such education take? Intensive, intermittent, or continuing? Structured classroom setting or one-to-one tutors? Extensive manuals and hard copy documentation or on-line prompts and HELP commands? And how should the system's use be expanded, assuming that this would benefit the organization? Through the educational processes mentioned above or by word-of-mouth? Can the diffusion of innovation, in this case a decision support system, be managed? All these questions, as well as the following hypothesis, merit research.

Hypothesis 6: *The success of a decision support system is directly related to the amount of education expended on its behalf.*

Organizational Issues

The dependence of decision support systems on their particular organizational setting makes them difficult to study.

What is important for Citibank, and works well for them, may not be at all appropriate for some other bank, let alone a nonfinancial institution. While inventory control systems and distribution systems may be generalizable, decision support systems tend to be unique. This fact came to the fore when Citibank attempted to develop MAPP both for internal use and for possible sale to other institutions. As MAPP developed, concern about external sale was increasingly overridden by internal requirements for the system (which is not surprising).

The continually changing nature of the organizational environment also affects decision support systems. At Citibank, the advent of Project Paradise had a profound effect on the need for a system like MAPP. Such reorganizations are a fact of organizational life and, to survive, decision support systems must be able to cope with them.

Personnel changes pose another organizational problem. For instance, not one bank officer who was interviewed in connection with the MAPP user study had been in the same job a year earlier. One manager, in fact, was in his third job in less than a year's time! How is such a shifting group of users to be supported? This makes the educational problem described in the preceding section doubly difficult. Surely the "half life" of even *successful* decision support systems is considerably shorter than that of their data processing cousins. These considerations lead to the following hypothesis.

> Hypothesis 7: *The success of decision support systems is inversely related to the organizational stability of the host environment.*

The last area for research addresses an issue that has been adroitly avoided so far. There have been several references to the "success" of a decision support system without any attempt to define "success." Although academicians can duck such thorny issues, MIS managers cannot, and they are being increasingly pressured to provide some rationale or justification for the decision support systems that they are sponsoring. "Improved managerial effectiveness" is of course the justification most often offered; and if the cost of the proposed DSS is small enough, or if the intended user is willing to pay for it regardless of the cost, then this answer is usually sufficient (though not very satisfying intellectually). Better measurements must be found; the final hypothesis posed for research is not so much a testable assumption as a challenge to the field.

> Hypothesis 8: *The measure of success of a decision support system is no harder—or easier—to gauge than measures applied to other forms of managerial activity.*

COMPUTER-BASED DECISION SUPPORT SYSTEMS: PROBLEMS OF DESIGN AND IMPLEMENTATION

Gennady B. Kochetkov
The USA and Canada Institute of the
USSR Academy of Sciences

INTRODUCTION

Since the end of the 1950s complex computer-based management systems for economic organizations have been developed intensively in the USSR. Separate ministries and agencies were responsible for this activity until 1965. Since then a national computer-based information gathering and processing system for planning, accounting, and control (GSCS) has been set up and continuously developed, as a part of the General State Plan. For a detailed analysis of different stages of computer implementation for national economic management, see Avtomatizirovannye sistemy upravleniia (1972), Glushkov (1974), Makhrov *et al.* (1974), Modin (1975), and Cheshenko (1977).

In accordance with the General Plan, 4,370 computer-based management systems (CMS) were in operation at the beginning of 1980, in most cases running on domestically produced computers. Of these, 1,649 systems have been designed for production control in the metallurgical, chemical, pulp and paper, energy, and communication branches, as well as other sectors of the economy. The remainder are computer-based systems for organizational management in ministries and large enterprises. At present, every national ministry and 30% of ministries at the republic level are equipped with computer-based management systems. The most complex interagency systems are functioning in the State Planning Commissio (GOSPLAN), the State Material and Technical Supply Committee (GOSSNAB), the Central Statistical Administration, and the State Committee on Science and Technology, to name a few. Regional computer centers in such large industrial centers as Riga, Tomsk, Tallin, and Tula have systems at initial stages of operation (Zhimerin and Miasnikov 1979, Zhimerin 1980).

A common theoretical and methodological framework was used in the design and development of all the computer-based management

systems. The compatibility of their hardware, software, and organizational structure has been given careful attention. As functional elements of GSCS, they must be capable of close interaction (Avtomatizirovannye sistemy upravleniia 1979, Borin 1972, Modin 1977, Upravlenie sotsialisticheskim proizvodstvom 1975).

Soviet management specialists define a CMS as a management system in which some functions or procedures are performed by computers (ASUP 1977). In general, a CMS contains the following major subsystems: economic, production planning and control, computer-aided design, inventory management, sales management, quality control, human resource management, finance, and auxilliary production control.

CONTRIBUTIONS AND SHORTCOMINGS OF TRADITIONAL COMPUTER-BASED MANAGEMENT SYSTEMS

As with other types of automation, the implementation of a CMS is only justified if it makes the work of managers more effective. Long-term experience with computer utilization for management tasks shows that efficiency has been achieved in two areas: (a) administrative expenses have been reduced; and (b) the quality of decision making has been improved (Informatsiia, kotoraia pomozhet rukovoditeliu 1980; Cheshenko 1978).

According to Soviet economists, decision makers use 30% of their working time to collect and process required information; staff workers use up to 80% of their time for this activity (Modin 1977). The first goal of computer-based management systems has been to relieve this work load and to reduce the expenses related to decision making, through office automation and rationalized allocation of resources. Only after greater efficiency was achieved in this sphere did CMS developers turn their attention to improving decision making procedures. Attempts to improve the efficiency of decision making through use of computer-implemented systems were first made in the mid-1970s.

Because of a number of constraints, computer-based management systems have been able to improve the efficiency of separate elements of management procedures, but not the efficiency of a management unit as a whole. One difficulty is tied to the diversity of experts working on such systems, and their lack of practical experience with management tasks. EDP specialists consider their goal to be the rationalization of the information flow structure and the use of information resources within a firm or company. MIS specialists concentrate on designing sophisticated information systems. Finally, MS specialists attempt to build models which may be useful to managers. Unfortunately, the majority of EDP, MIS, and MS specialists do not have managerial experience; they do not understand the nature of the management profession, and therefore analyze it only in terms of general formal attributes. Each group of specialists also tries to protect its own interests in the course of designing computer-based management systems. For these reasons, most of the systems that have been built recently have not satisfied the requirements of managers for decision making and have thus been used only occasionally.

THE TRANSITION TO DECISION SUPPORT SYSTEMS

All the traditional approaches mentioned above (EDP, MIS, MS) attempted to promote the use of computers in the sphere of decision making. However, practical experience shows that most decisions are made hurriedly in tense situations, and therefore are difficult to predict. For this reason seemingly rational and efficient EDP systems have proven useless in such situations. Another problem is that the models that have thus far been elaborated are conceptual in nature; they are not intended for use in specific decision-making situations. Even if a manager has access to an efficient computer system, he needs the assistance of a programmer. Finally, each business situation is unique, and decision making is a creative process. A manager cannot specify in advance the kind of information he will need, for managerial decision making deals with exceptions rather than rules.

A way out of this situation is only possible if managerial activities and decision-making tasks become an integral part of computer-based management systems. This concept underlies recent work in the area of decision support systems (DSS). Such systems require that managers as well as general staff work out alternative decisions and analyze their implications; as man-machine systems become operational, managers can interact with the computer to generate alternatives and make choices.

Although computers have long been considered by Soviet scientists and administrators to have great potential for aiding managerial decision making, implementation of this goal has been difficult. It was necessary first to build an appropriate organizational, economic, and social infrastructure. At the present time this work is in its final stages. A satisfactory hardware base is available, many different models have been designed, and the necessary software has been developed. Also, a concerted effort has been made to retrain personnel. This work, together with previous experience with computer-based management systems in ministries and enterprises, has made it possible to build decision support systems.

More and more managers are coming to the conclusion that computer-based management systems should not only be used to rationalize use of resources, but also to aid in decision making. Specialized systems have already been created for top-level managers in a number of enterprises for this purpose. "Apparat" was one of the first systems of this kind (Tychkov 1978). A preliminary version of this system was put into operation in 1978. Work is now in progress to develop a subsystem for economic analysis of business situations. Interest is building in this activity and many enterprises have started building their own systems (Informatsiia, kotoraia pomozhet rukovoditeliu 1980).

CONCLUSIONS

Although active implementation of decison support systems is just beginning, earlier experience with computer-based manage-

ment systems and recent DSS experiments permit certain conclusions to be drawn.

(1) The growing interest in problems relating to the design and implementation of DSS in the Soviet Union implies a major shift in the thinking of managers. They are no longer just interested in increasing the efficiency of computer-based management systems, but wish to improve economic organization as a whole. The transition from traditional CMS to DSS marks a new, more sophisticated stage in the application of computers for management purposes. Available evidence indicates that organizations with long-term experience with EDP, MS, and MIS approaches are most successful in designing and implementing DSS.

(2) A choice can be made by a human being, animal, machine, etc. However, a social choice can only be made by a human being; it depends on previous social experience and future actions, and is based on the emotional background of the individual. Social choices may be influenced by elements that do not pertain just to the subject-matter of the problem at hand, but also reflect a number of environmental factors—time, order, frequency, speed of events, etc. (Tikhomirov 1972, Tikhomirov 1973). The social nature of choice must be taken into account by DSS developers.

(3) In practice managers are often affected by the factor of uncertainty, for they cannot know which particular problems will arise. Moreover, the specific features of decision making in real organizations differ radically from theoretical formulations (Vishniakov 1972, Organizatsionno-pravovye problemy ASU 1979). As a rule managers cannot explain in detail how they make decisions in practice, for so many elements of decision making are informal and socio-behavioral in nature. Behavioral factors must be taken into consideration in the design of DSS and special research is required in this area.

(4) Man-machine interactions in the decision-making process is a problem which has scarcely been explored. For the computer to amplify the manager's intellect, DSS hardware, software, and 'people ware' must be sophisticated enough to aid real decision making. At the same time, managers must be open-minded about decision support systems and motivated to interact with the computer. The availability of interactive terminals, microcomputers, and special hardware is of great importance here. The opportunity for free interaction between manager and computer is a condition for the effective operation of decision support systems. (We must admit that in practice the majority of such systems do not provide direct contact between manager and computer.)

(5) A DSS can be especially effective when applied to team decision making—i.e., decision making by committees, boards, task forces, etc. Each member of a team may ask the DSS a "What if" question, and the answers can then become the subject of team discussions. Team decision making is of course quite different from individual decision making, and this must be taken into consideration in the design of decision support systems.

(5) At present computerized management systems in industry rely on a rather large number of optimization models. A model base has been elaborated that can perform such functions as long-range planning, production development, plant location, and national economic planning. DSS are expected to be greatly effective in this area (Kanygin 1980, Tychkov 1978, Cheshenko 1978). However, the application of DSS in strategic planning and long-range forecasting will require radical modification of the basic concepts underlying computer implementation in management. Until now computer-based management systems have been oriented toward resource optimization; the tasks and functions in such systems are well-structured, repeated periodically, and can be typified. In contrast, each of the tasks of strategically oriented computer-based systems is unique.

As shown in this paper, Soviet managers and scientists consider managerial decision support to be a goal of highest priority. The concept of DSS has become an integral part of the national computer-based system for planning, accounting, and control. For this reason many governmental and industrial organizations are busily engaged in designing decision support systems.

REFERENCES

ASUP: Obshcheotraslevye rukovodiashchie metodicheskie materialy po sozdaniij avtomatizirovannykh sistem upravleniia predpriiatiiamy i proizvodstvennymi ob"edinenijami (1977) (Methodological works of a general industrial branch; managerial perspectives regarding the creation of automated control systems for industrial enterprises and production complexes, in Russian) Moscow: Statistika.

Avtomatizirovannye sistemy upravleniia (1972) (Automated control systems, in Russian) Moscow: Ekonomika.

Avtomatizirovannye sistemy upravleniia: itogi i zadachi (1979) (Automated control systems: results and aims, in Russian) EKO 2: 32-59.

Borin, V.G., ed. (1972) Aktual'nye problemy upravleniia (Today's managerial problems, in Russian) Moscow: Znanie, Vol. 1.

Cheshenko, N.I. (1977) Nekotorye problemy razvitiia ASU v desiatoi piatiletke (Some problems in developing automated control systems for the tenth five-year plan, in Russian) Ekonomika i matematicheskie metody 5: 1085-1092.

Cheshenko, N.I. (1978) Otsenka effektivnosti sozdanija ASU (Evaluating the effectiveness of automated control system development, in Russian) Moscow: Statistika.

Fedorenko, N.A. (1968) O razrabotke sistemy optimal'nogo funktsionirovaniia ekonomiki (Concerning the development of systems for an optimally functioning economy, in Russian) Moscow: Nauka.

Glushkov, V.M. (1974) Vvedenie v ASU (An introduction to automated control systems, in Russian) Kiev: Tekhnika.

Glushkov, V.M. (1977) Industriia pererabotki informatsii (The information processing industry, in Russian) Kommunist 12: 41-50.

Informatsiia, kotoraia pomozhet rukovoditeliu (1980) (Information that aids the manager, in Russian) EKO 1: 96-114.

Makhrov, N.V., A.A. Modin, and E.G. Iakovenko (1974) Parametry razrabotki sovremennykh avtomatizirovannykh sistem upravleniia predpriiatiem (Parameters for developing modern automated control systems for industrial enterprises, in Russian) Moscow: Nauka.

Modin, A.A., N.Ia. Petrakov, Iu.I. Cherniak, and E.G. Iakovenko (1975) O nekotorykh tendentsiiakh razvitiia ASU (Certain aspects of automated control systems, in Russian) Ekonomika i matematicheskie metody 4: 619-627.

Modin, A.A. (1977) Razvitie organizatsionnykh struktur v avtomatizirovannykh sistemakh upravleniia (Developing organizational structures in automated control systems, in Russian) Ekonomika i matematicheskie metody 6: 1164-1174.

Organizatsionno-pravovye problemy ASU (1979) (Organizational-legal problems of automated control systems, in Russian) Moscow: Nauka.

Tikhomirov, Iu.A. (1972) Upravlencheskoe reshenie (Managerial decisions, in Russian) Moscow: Nauka.

Tikhomirov, O.K., ed. (1973) Chelovek i EVM: psikhologicheskie problemy avtomatizatsii upravleniia (Man and computer: psychological problems of control automation, in Russian) Moscow: Ekonomika.

Tychkov, Iu.I. (1978) Rukovoditel' i ASU (The manager and automated control systems, in Russian) EKO 5: 100-112.

Upravlenie sotsialisticheskim proizvodstvom (1975) (The management of socialist production, in Russian) Moscow: Voprosy teorii i praktiki, Ekonomika.

Vishniakov V.G. (1972) Struktura i shtaty organov sovetskogo gosudarstvennogo upravleniia (The structure and staffing of Soviet state administrative organs, in Russian) Moscow: Nauka.

Zhimerin, D.G., and V.A. Miasnikov (1979) Avtomatizirovannye avtomaticheskie sistemy upravleniia (Automated automatic control systems, in Russian) Moscow: Energiia.

Zhimerin, D.G. (1980) Soveremennye real'nosti ASU (Automated control systems today, in Russian) Pravda, May 12, 1980, p. 3.

AN INTERACTIVE MODELING SYSTEM FOR ANALYSIS OF ALTERNATIVE DECISIONS

Victor V. Gelovani
Vladimir B. Britkov
Valentin V. Yurchenko
All-Union Research Institute for Systems Studies

INTRODUCTION

In order for computers to be useful to managers for decision making, both models and tools for interaction between managers and models are needed. To create practical tools, several difficulties must be overcome. First, the creation of general models covering multiple aspects of given phenomena is often an insuperable task. Therefore it is often desirable to create special models on a case-by-case basis. Software modeling languages are needed to facilitate this process; a number of languages, such as Dynamo, can be used to create models of simple systems, but they are not useful for modeling large, complex systems. For such cases models need to be assembled from separate submodels and units. It is useful to prepare a library of submodels, from which relevant blocks can be chosen to treat multifaceted problems. This task has been undertaken with the help of computers at the All-Union Research Institute for Systems Studies. An interactive regime is used, with human participation.

The development of scenarios, as a formal method for taking into consideration the intuition of managers, is another problem area under study at the Institute. Scenarios provide managers with an opportunity to introduce their own knowledge and assumptions into the modeling process. Scenarios can be simple or complex, multilevel and hierarchical in structure; it should be possible for managers to create a scenario during a session on a computer terminal.

Institute staff are also trying to resolve the difficulty posed by cases in which a scalar criterion is lacking for the complex object under study--especially for social factors. Numerically different variants or decisions are difficult (and sometimes impossible) to compare. This problem cannot always

be solved with the help of a vector criterion. In some cases it can be transformed into another type of problem, sometimes a more difficult one--e.g., the problem of determining the influence of every element of the criterion vector. For dynamic systems it is also difficult to choose the moment in time to consider the criterion.

AN INTERACTIVE MODELING SYSTEM

An interactive modeling system is being developed for the analysis of alternative decisions at the All-Union Research Institute for Systems Studies; it is intended for building, testing, and using models for decision-making purposes. It provides for model assembly, parameter identification, sensitivity analysis, implementation of various scenarios, usage of models in simulation and optimization modes, and interpretation of results. Fortran is used to run submodels of this system; it is widely known and the software of every computer can be adapted to this language.

The modeling system consists of several elements, as shown in Figure 1. A central element is MIM (Monitor of Interactive Modeling). It permits researchers to interact with all the other elements of the system by means of a display keyboard. Models

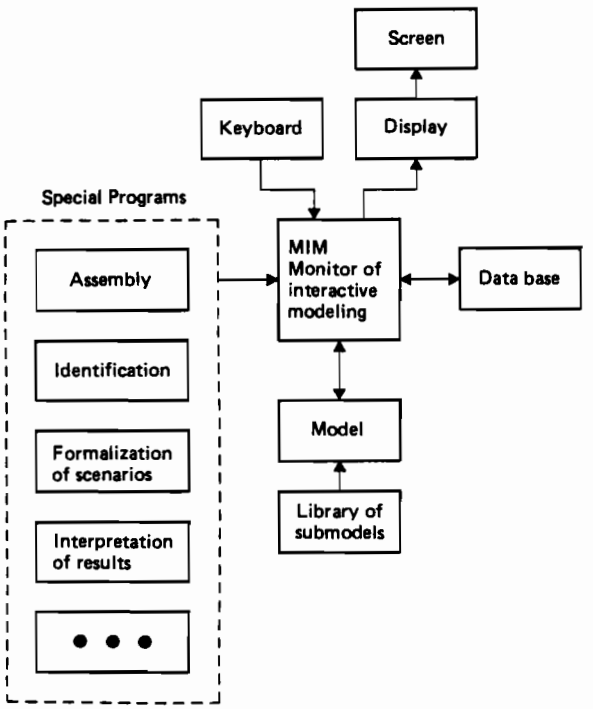


Figure 1. The structure of the Interactive Modeling System.

consisting of linked submodels, constitute another element of the system. The models and the submodels are Fortran subroutines, which permit the solution of a system of differential equations. Yet another element is the data base management system, which facilitates the identification and aggregation of data. As well, special system programs provide for the assembly of a model from its submodels, the formalization of scenarios, and the calculation of model parameters on the basis of data from the data base. There are also programs for sensitivity analysis and interpretation of results. All the programs operate in an interactive mode.

After a model has been built, a decision maker may begin to investigate alternative decisions. He can introduce various assumptions with the help of a scenario formalization program and then perform model runs. He is subsequently able to obtain the results of different runs from the disc memory and to compare them. The results of the modeling exercise can be represented on the screen in the form of graphs and tables. In the case of a group of decision makers (decision committee), a large screen with a projector is often used. The decision maker studies the results (or the members of the committee discuss the results). Afterwards, the decision maker may introduce new assumptions and control decisions to the computer system and repeat the above procedures. This interactive modeling system is currently running on a PDP 11/70 computer at the Institute and its development is continuing.

THE STRUCTURE OF DECISION SUPPORT SYSTEMS

Roman L. Sheinin
All-Union Research Institute for Systems Studies

Improved computational abilities, decreased size, improved reliability, simplified maintenance procedures, and abrupt decreases in cost have led to the wide use of mini-computers for management tasks in the USSR. Previously the use of mini-computers in management was limited to routine and repetitive tasks and to very specialized, time-consuming tasks, such as long-range forecasting of consumer demand for goods.

Organizations as a whole profited from the use of mini-computers to perform routine tasks; however, this type of application generally has not been of help to top-level managers. Although applications of mini-computers to carry out highly specialized tasks had potential for aiding managers, it was very hard to resolve unexpected problems and thus to support many activities carried out by managers. The goal of Decision Support Systems is to assist in overcoming these drawbacks. Awareness of the necessity of Decision Support is becoming more widespread. We hope that the development of DSS will lead to more use of computers in management in the USSR.

APPROACHES TO THE DESIGN OF DSS

At the present time a number of Decision Support Systems already exist, and they are of assistance to managers. Still, the field is at an initial stage of development and analysts have many different views about the role and position of DSS within organizations. For the purposes of this paper, the following assumptions may be made:

— DSS are being designed for direct help to top-level managers.
— The main activity of top-level managers involves decision-making processes.

— The most complex and crucial decision-making processes, in terms of organization stability, are those concerning unforeseen and ill-structured problems.

Thus, the aim of DSS design is to provide effective direct assistance to top-level managers when they make decisions about previously unanticipated and ill-structured problems. This is the principle aim of DSS, though surely not the only one. To achieve this aim DSS should have at least two classes of capabilities: (1) the capability to structure ill-structured problems and (2) a means that allows managers to respond instantly to unforeseen problems.

We can differentiate two approaches to the design of DSS, namely, a universal approach and a problem-oriented approach. The first is based on decision-making theory and the theory of choice. It is assumed that in the decision-making process a manager faces a set of alternatives. Each alternative can be associated with some characteristics that describe its quality. If these characteristics can be specified, formal methods can be applied to determine the best alternative. Although it is often difficult to identify such characteristics, there is hope that the methods of performing this task will improve, and that such methods can be integrated into particular Decision Support Systems.

The universal approach to designing DSS has serious drawbacks. It often happens in management practice that a manager has a problem, but not a set of alternatives. First he has to determine several courses of action to resolve his problem and then to choose the best one. In other words, the decision-making process consists of two stages, generating alternatives and choosing between alternatives. The problem-oriented approach rather than the universal approach should be applied in this case. In the problem-oriented approach a peculiar means is generated as a given problem arises to assist in the process of generating alternatives and resolving the problem.

THE STRUCTURE OF DSS SOFTWARE

One can represent the structure of DSS software in terms of the following components: (1) data base; (2) the language of interaction between a user and DSS; and (3) specific software that provides DSS with "intelligent" capabilities. The data base is used for storing information on past organizational activities and on some relevant aspects of the organizational environment. The language of interaction between a user and a DSS should allow relatively simple and easy dialogue. Specific software is in some ways the kernel of DSS, for the effectiveness of DSS depends critically on this software. Such software should consist of at least the following parts:

— Packages which allow the structuring of arrays of data. (These packages make it possible to use such methods as factor analysis, cluster analysis, pattern recognition, multi-dimensional scaling, and so on.)

— Forecasting packages.
— Packages for handling the opinions of experts.
— Packages for simulating the behavior of an organization and its environment.

It is possible to use the first type of package as building blocks during the process of designing packages of other types. In some ways packages that aid in the structuring of arrays of data are basic to DSS.

ISSUES FOR THE FUTURE IN DSS:

Report of Discussion Group 1

Gary Dickson
University of Minnesota

Following the introduction of its members, Group 4 began unconstrained brainstorming about major issues in the area of Decision Support. It soon became apparent to nearly all group members that a definition of Decision Support was required to deal with some of the issues. A major breakthrough occurred when it was recognized that a process exists that leads to the production of a specific Decision Support System. Much confusion is caused when the title DSS is applied both to the process and to the resulting product. After much discussion, the following definition evolved:

- Decision Support Engineering (DSE) has as its primary product a Decision Support System (DSS).
- DSE is a practice, as are management, medicine, and law.
- Like other practices, DSE draws upon a number of disciplines and technologies. Computer science, operations research, organizational behavior and management, economics, philosophy, political science, and organizational functions are important examples.
- The primary purpose of DSE is to improve decision making.
- DSE is different from operations research (and management science); its product is a dynamic, adaptive, and interactive process involving the decision maker, rather than a relatively static model.
- A DSS is different from an information system (IS), for it contains evaluation-aiding capabilities such as optimization models, utility-indicating displays, or guidelines for structuring problems.

The group felt that the word "engineering" in the term DSE may be an unfortunate choice because of its connotations to users and managers. Decision Support Analysis (DSA) may be a better term to apply to the process.

The above definition raised several issues, such as whether DSE and DSS differ from previous practice and whether DSE is a "pure" discipline. Many issues remain unresolved. The approach chosen by the group was to identify issues on two levels. The "Meta Level" deals with global issues associated with the practice of DSE; their resolution will be directed toward improvement of the practice. The "Micro Level" treats issues associated with building a particular DSS. It should be noted that:

— The issues discussed below are illustrative and certainly not exhaustive.
— The issues are not given priorities.
— Many of these issues will be resolved through a process of evolution. The process will be as follows: Issue identification→Research practice→Evaluation→Communication to the practioner→Practice→Evaluation→Back to research practice.

META LEVEL ISSUES

Identification of Actors and their Relationships

It is important to define the parties involved in DSE and to specify how they should relate. Sprague provides the notion of the Manager/User, the System Builder (the DSE) and the Toolmaker. Toolmakers provide technology and practice to the two other groups. They consist of "scientists" (e.g., computer technologists, pure behavioral scientists) and "generalists." The latter are typically found in business schools and know "something" about many areas, such as those listed in the DSE definition. The generalists may in some cases also be builders, but more often they provide instruction or facilitate the building process. Scientists typically work for government, institutes, universities, or technology vendors. The vendors (the State in non-market economies) make DSE technology available.

Analysis of Requirements

Historically the scientist toolmaker has defined the characteristics of the tools with which the builder and the user must work. In other words, builders and users have been technology-driven, giving little input to the creation of DSE tools. To avoid previous mistakes, DSE must help define conceptually and through research the requirements to be met by DSE tools. Requirements would include:

— Tools, e.g., Database, Artificial Intelligence, Hardware, Natural Languages.
— Processes, e.g., Individual Difference Handling, Role of Decision Processes.
— Evaluation Methods, e.g., Case Studies, Experiments, Longitudinal Studies.

Performance Measures

It is crucial that methods for measurement be developed, so that the quality of a DSE process and a DSS can be evaluated.

Problems of Theory and Definition

The definitions of DSE and DSS need to be refined and agreed upon by users, builders, and toolmakers. It must be acknowledged that at present there is no "theory" of DSE, i.e., it is not a pure discipline with an established knowledge base. Despite this condition, we should not wait to practice until a theory is developed. We can draw upon theory and knowledge from the support disciplines identified above and work iteratively to develop our own "theory of DSE." After all, IS/DP have only existed for a quarter of a century and DSE only for a few years. We should acknowledge that our understanding and tools are still primitive. However, the potential benefits and needs are too great to permit inactivity because of such a lack of full understanding.

Impact on Organizations

The degree to which decision support engineering and the resulting decision support systems provide for organizational integration should be explained. Further, their effect on the centralization/decentralization of decision making and organizational functions must be addressed.

MICRO LEVEL ISSUES

Development and Teaching of the DSE Process

There exists a need to develop the DSE process and to teach this process to DSE practitioners and user/managers.

The User/Builder Interface

The two-way flow of influence between the DSS builder and user must be explored. The results ought to be reflected in our DSS building practices. Specifically, the issue is, To what extent should each party determine the nature of the DSS? Keen describes a two-way relationship, but the group felt that the role of the user may be overemphasized—perhaps as a reaction to the OR experience. We are concerned that the result may be a number of "acceptable" decision support systems that support poor decision processes (efficient but not effective). As Flores stated:

> ...the common belief is that design should be dominated by the desires of the *user*. Prejudices derived from this belief are (a) that the best way to discover what the user wants is by asking questions by means of

interviews and by observing and studying the user, and (b) that the criteria for design will evolve inductively after a trial and error process. We believe this approach is wrong.

To simply implement the user's desires is extremely questionable. However, a builder-dominated process must be accompanied by sound implementation strategies and practices.

Overcoming Managerial Resistance

Building technically sound but unused decision support systems is not cost/beneficial. A *socio*-technical perspective is important. Operational implementation practices are needed as well.

Flexibility and Modular Packages

DSS designs must provide for adaption to individual differences and environmental conditions. We assert that DSS toolmakers

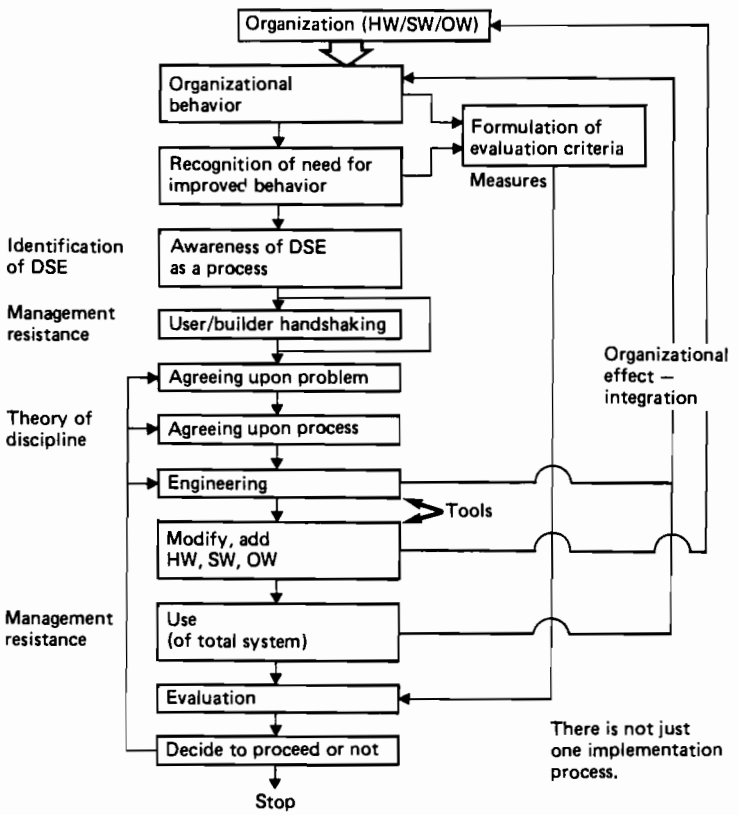


Exhibit 1.

must create more flexible and modular tools than in the past. Analysis of requirements ought to confirm this assertion.

Exhibit 1 has been prepared to summarize our view of the DSE process. A model of the DSE process is shown, in which some of the issues are mapped to the process.

ISSUES FOR THE FUTURE IN DSS:

Report of Discussion Group 2

Leif Methlie
Norwegian School of Economics
and Business Administration

A DSS PERSPECTIVE

In order to understand the findings of the group in terms of research issues and an action plan, it is first necessary to clarify the group's conception of DSS. DSS is viewed as an *application concept*, i.e., it is mission-based and implies applying technology to tasks through human beings (the users). This view emphasizes behavioral factors and the *process* of design and implementation. Technology is the means, not the end.

Other perspectives discussed were (1) a more normative structure-based perspective, according to which a DSS is determined in terms of the structure of the tasks to be supported, and (2) an information-based perspective that excludes decision making and emphasizes information requirements.

CHARACTERISTICS OF DSS

The term "decision" in DSS may be misleading. First, it is often hard to identify the decision; second, managerial or intellectual work is seldom viewed in terms of decisions, but rather in terms of activities like meetings, mailing, and telephone calls. A DSS should be thought of as *supporting decision making*, including intellectual activities performed by managers, doctors, lawyers, and other professionals. "Decision support" is considered a narrower view of DSS than "support for decision making."

The following were identified as key characteristics of DSS:

(1) *Underspecification*. The degree of underspecification of functional properties of a DSS is very high. This may not be unique to DSS; it is probably true in any computer processing

system. However, in transaction processing systems underspecification occurs because the system builder knows less than the user about the way in which the system is to be used, i.e., it is difficult to capture the requirements. In the area of DSS applications, it is impossible to determine the requirements.

(2) *Adaptivity*. An adaptive DSS means that the system can adapt to the user's changing requirements. Thus, if we start with a specific set of tools (hardware/software) it is likely that various users will apply the tools differently. Therefore the tools will evolve into different types of DSS—one unique DSS for each user. Two aspects of adaptivity can be identified: (a) *learning* and (b) *a changing environment* (unforeseen or changing problems). The DSS should not only adapt to changing requirements of the user as he/she learns to utilize the available facilities, but should also *stimulate* learning. Note, however, that the learning aspect of DSS is not similar to computer-aided instruction (CAI) where the knowledge transferred to the user can be prespecified (and the answers to the problems are known).

(3) *System behavior*. The behavior of the system is driven by the user. Thus in DSS the linkage between user and system is very strong. This helps to differentiate a DSS from the traditional MIS; in MIS the strong link is between task and system (a system is designed for a specific task). This can be illustrated by the following figure:

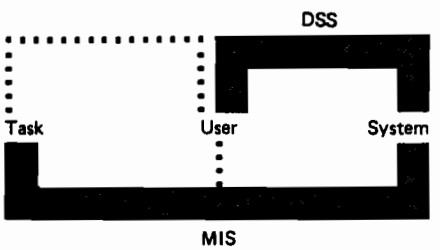
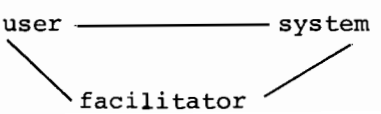


FIGURE 1 Strengths of links between task, user, and system in MIS and DSS.

(4) *Facilitator*. In an application perspective of DSS, where the emphasis is on the process of applying technology more than on a specific technological product, the role of facilitation seems to be extremely important. We will therefore introduce the concept of a facilitator, which can be linked to system and user as shown:



The facilitator can take on different roles in the implementation-design process of a DSS. One role focuses on the user—i.e., facilitates the identification of a target (a task and a user) and getting started. This is the 'change agent' role. Its main focus is on implementation. The second important role of the facilitator is to focus on the system—i.e., the system building aspects. A facilitator can be a single person (such as a manager experienced in using DSS tools or a systems designer building a specific DSS) or a group of persons.

PROBLEM AREAS

A number of problem areas are listed below. The list was produced through a process of brainstorming. It is in no way exhaustive, and each item is accompanied only by short comments. No priority is given to any of the items.

— *Implementation*. In an application perspective of DSS this implies pre-design as well as post-design problems (i.e., how to get started and how to evolve).
— *Descriptive aspects of DSS*. This includes an understanding of and methodologies for studying user tasks, user-system behavior, user-facilitator interdependencies, facilitator-system interdependencies, and the organizational context.
— *Representation of tasks*.
— Need for a *normative model* of user effectiveness (i.e., What does it mean to have a good system?) In OR the normative model very often is an optimization model. In an adaptive DSS the normative model may be difficult to define, thus making benefit assessments and a posteriori evaluation difficult.
— Benefit assessment as an a priori evaluation. This includes short-term and long-term benefits. The problem here is to identify and measure benefits.
— A posteriori evaluation, involving definition of criteria (pre-design aspect) and measurements.
— Classification and evaluation of the DSS task. Included in the DSS task are tools for display management, data management, model management, communication, and interfacing. Tasks must be evaluated in relation to user behavior.
— Lack of design architecture.
— Philosophy and topology of communication.
— Social impact of decisions.
— Normative theory of ergonomics.

ACTION PLAN

Implementation Research. How do we prove (demonstrate) that a DSS is *feasible*? This problem may be subdivided into the following questions: How do we find an appropriate target (task), a client (user), a facilitator, and a DSS-development methodology (e.g., prototype)? Can we identify characteristics of a "good" facilitator for certain types of clients and tasks, and is the development methodology contingent on type of facilitator, task, and client?

So far the *research methodology* applied in the DSS field has been dominated by case studies. The case study approach is explorative and descriptive. Lack of prescriptive theories in this field is probably due to this concentration on case studies. Implementation research based on comparative studies in controlled environments probably constitute a more appropriate research methodology. Even laboratory settings can be used to some extent (restricted to problems that can be studied out of context, e.g., graphs versus tabular presentation of information). Because DSS is an application-oriented field, research and development can sometimes be difficult to separate. Greater care should be taken to separate research methodology from development methodology.

Evaluation Research. Evaluation research encompasses three problem areas: developing a normative model of user effectiveness; benefit assessment; and a posteriori evaluation. Key points here are (a) *What* do we evaluate—user behavior, user-system behavior, the output (decision), or the decision-making process? and (b) *How* do we measure it—in terms of usage of the DSS functions, by the rate of change in usage, or by diffusion of the DSS (generator) into multiple DSS-specifics? Can we develop metrics? Is a panel of experts appropriate?

Tools Research. Can we classify users into classes in order to create DSS generators? This issue must be explored from several angles:

— Tasks (functions). The DSS generators in existence today are structured according to business functions and most are found in the financial analysis and planning area.
— Organizations. Are different DSS generators dependent on organizational characteristics (public, technically advanced, industrial, etc.)?
— User behavior (cognitive styles, etc.).

Can we come up with a model of a class of users independent of task and organizational context?

To create a *framework for research*, we must ask questions such as the following: Can we recognize patterns of usage of DSS generators for particular users, tasks, and organizational characteristics? Will generators focus on different aspects, such as modeling (analytical capabilities), data, text, and communication? A task group should be set up to define and formulate these questions in more precise terms in order to promote research in this area.

Committee Support System. As most DSS research has been oriented towards individual decision making and personal tools, the group found it very important to devote more research to committee activities. The group recommends that a future conference be devoted to this topic.

FINAL REMARKS

The findings presented above are based on the many interesting problems/issues/questions raised during the discussion. However, this report is the Chairman's interpretation of the discussion and he takes full responsibility for the interpretation and presentation.

ISSUES FOR THE FUTURE IN DSS:

Report of Discussion Group 3

G.R. Wagner
Execucom Systems Corporation

It was indeed a challenge to chair this discussion among multidisciplinary scientists from different countries. Their different views, perspectives, and scientific and social backgrounds have contributed to the knowledge base needed for understanding decision support systems (DSS). In this brief paper it is not possible to do justice to the many ideas expressed; I hope only to capture the primary contributions affecting issues, questions, and actions for the future.

After discussing at length whether DSS should be defined, we decided that it was premature to suggest tight boundaries in the form of definitions. However, our discussions touched on the important issues that arise when academic researchers identify themselves with a field that lacks a theoretical foundation. The group decided to use Figure 1 to portray DSS and to address this important issue.

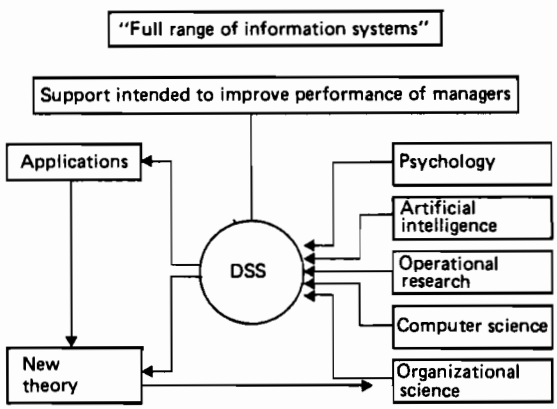


FIGURE 1

The mythical full range of information systems, which has the intended benefit of supporting managers, is represented at the top of the diagram. The subset called DSS falls within that full range of information systems. Disciplines contributing to DSS include psychology, artificial intelligence, computer science, operations research, and organizational science. These contributing disciplines provide DSS with the technology for building applications intended to influence and improve management decision making and to support intellectual activities generally. In this process new theories are identified through applications. DSS is therefore a link for various source, reference, or contributing disciplines to gain access to management and to identify new theories experientially. Thus we felt that there is adequate support for researchers from contributing disciplines to become involved in DSS and that there is ample belief in the long-range viability of DSS for researchers from various contributing disciplines.

We then identified specific areas in which there were questions and problems; the remainder of this paper summarizes the group's progress, beginning with general issues and proceeding to specific technical problems.

There is clearly a need to increase overall awareness of DSS at all levels: managers, builders, researchers, and others. Evidently practitioners of DSS sometimes lack awareness of the work of DSS researchers and vice versa, and various sectors misuse the term *decision support systems*. It is thus important to create awareness through conferences, such as the College on Information Systems workshop in August 1980 and the conference planned for Atlanta in June 1981, and through academic and trade journals. The publication of case histories, though essential, is rare and should be promoted. We should also be taking advantage of the opportunity to clarify and promote DSS concepts and contributions, for it is not unusual for mass media authors who use the words *decision support systems* to be unaware of the spirit and characteristics of DSS.

Another important issue is developing a means to recognize a DSS, certainly an important prerequisite for general awareness. In practice, collecting data to document implementations would contribute to this goal of a "check list." We need assistance in answering the question, "How do I know whether what I have is a DSS?"

A profile of DSS tasks and methodologies is also desirable. This taxonomy might simply list decision and planning situations or tasks for which DSS is (or is not) useful and methodologies that have (or have not) proved successful in those situations. While such a document lends itself to obsolescence and even to error, it could be helpful.

We also identified several specific technical problems that we view as being addressed by a synthesis of source disciplines. For example, what factors contribute to managers' commitment to their DSS? We see a need for research regarding

what managers do. Perhaps this research should be of a different type than that published by Mintzberg and others; rather than collecting data on what the manager does now, we might collect data on what a manager could do differently if given electronic support. We recognized the need to document case studies that identify the barriers to success and the DSS "champions" in successful situations. It is also necessary to identify the characteristics of the environment, of the DSS task, and of the DSS methodology that are needed for a manager to feel satisfied and comfortable with, as well as committed to the DSS as part of his normal working activity.

How do we measure the performance of a DSS? Answering this question can involve a variety of measurements, ranging from highly subjective to sophisticated cost/benefit analyses. Whether performance is measured on the basis of the owner's happiness or a more complex cost/benefit analysis is not necessarily the issue; rather, the issue is to assure that an effort is made to measure the performance contributions of a DSS.

An interesting discovery during this conference was recognizing our lack of understanding about adult learning. Some advocate that the most important contribution of a DSS is to assist the owner in understanding and learning more about his environment; everyone would believe this to some extent. Given this contribution of a DSS, the need to incorporate adult learning theory into the DSS building process becomes important. Once we understand the learning process, we must develop measures to determine how well a DSS supports the task of adult learning. Research in this area is interdisciplinary and includes developmental and cognitive psychology, education, and cognitive sciences in general.

In the interest of flexibility, adaptability, transportability, and user acceptance, it is essential that a DSS be documented adequately. There is a clear need for technology to support documentation, such as self-documenting languages, text generation from program code, and user-oriented representations of data and models. If the user is to believe and to give credibility to his DSS, he must understand what is "inside it." In order to do that, he must be able to understand the logic, data, and origin of answers.

A particularly interesting and potentially important issue was how to gradually shift dialogue initiative from user to system. A relevant way to think about this problem in practice relates to asking "What if ...?" A user sitting at a terminal for hours may ask so many "what if" questions that he becomes literally lost—unable to cope with the amount of information being provided to him. It seems reasonable that part of the burden of the user at the terminal can be transferred to the system; this includes such activities as scenario generation, hypothesis generation, hypothesis testing, and even the adaptive guiding of the evolution of a DSS. This sort of movement is important in the next generation of DSS, where more intelligence is built into the system to relieve the amount of dialogue required of the user and to provide him with better summary information.

Thus far most Decision Support Systems have been so-called "personal decision support systems." Clearly, the need for group and organizational DSS, with which we have had little experience generally, is going to grow. We may face an organization and communication problem in order to integrate the three DSS levels. Issues of compatibility and interconnection from individual DSS to group and organizational DSS could be a major problem, particularly with the proliferation of personal computers.

Although much discussion centered on database management system technology, most DSS are used to look into the future; as such, they contain a significant percentage of subjective or judgmental data. DSS researchers—and to some extent DSS practitioners—have not addressed serious questions about how to accumulate and manage such data. These include such questions as how we obtain data from people, how we flag the source and propagation of this data, and how we ascertain its quality.

The contributions of this working group are clearly only a beginning; each of the areas identified could result in major research projects. However, we believe that the topics identified represent potentially significant contributions to furthering the base of knowledge related to successful DSS. Our audience ranges from the executive, who needs information to support his daily activities, to the academic researcher, who has a professional and personal need to contribute original research.

ISSUES FOR THE FUTURE IN DSS:

Report of Discussion Group 4

Richard Hackathorn
University of Colorado

The intellectual and emotional intensity of Group 4's discussion makes it difficult to convey the experience within a written description. It must also be stated at the onset that the group considered its mission (specification of the aims and program for DSS research) inappropriate and rejected it.

Despite the above qualifications, the group arrived at several conclusions that it felt were worth sharing. First, this conference and other activities related to DSS are part of an extremely important social movement; it is spurred by a common dissatisfaction with the application of technology to managerial decision making. This frustration, rather than the intellectual base of a clear theory, provides us with a sense of community. The theory simply does not exist!

The second consideration is that a practical theory for Decision Support will not emerge until we are willing to deal with more profound conceptions of human decision making. Each of us should take care to become aware of the cultural limitations inherent within a "technical" or "engineering" orientation. Each of us should recognize, with deep humility, that our fundamental values may tragically disable our honest mission of improving managerial decision making.

To convey these ideas, the group formulated the following modest proposal:

> We believe that managers live in a constant state of transition. Perplexity is always within the manager's mind, and this will not change. The manager will continue to act without full understanding and

> will not consider this to be a problem; while attempting to increase his understanding, he never expects to arrive at a full understanding.
>
> We do not want DSS to be another "MIS." That was a mechanistic attempt at a "solution," which does not and never did work. We must take another approach.
>
> We recognize that any attempt to give definite form to DSS would be an attempt to deny the necessity of living with perplexity. We prefer to be a *movement* with no attachment to any technique. We choose to act from a sense of this movement, not because of an attachment to a technique. And we choose *not* to act in ways which imply knowledge that does not exist.
>
> Therefore the essence of this proposal is that the issue before us is an ethical rather than technical one.

In conclusion, the group strongly recommends that future activities related to DSS research have a component that seriously deals with the topic "Ethics of Decision Support Intervention."

ISSUES FOR THE FUTURE IN DSS:

Integrating Session Summary

Michael A.H. Dempster
International Institute for Applied Systems Analysis

INTRODUCTION

This statement is an attempt to summarize the presentations and discussion of the integrating session of the Task Force Meeting on Decision Support Systems (DSS). Although summarizing is never a particularly easy job, a number of important points have emerged clearly from this meeting. I do not intend to give an exhaustive list of these points, and my emphasis is of course personal, as it must be in any summary.

I cannot overemphasize the point made by Group 4: we are here because we believe that an important social movement is going on. IIASA has begun research in the general area of information technology for this very reason.

We must also recognize, as Flores and several others pointed out in both public and private discussions, that there are limits to rationality. We never have—and can never have—complete descriptions of things; science is trying simply to improve things, at least marginally. We must recognize that managerial endeavor, as any kind of human endeavor, is conducted with perplexity in the face of complexity.

On that note we must caution ourselves that humility is necessary in forwarding this general area of professionalism. There was an implicit call at this meeting for a professional ethic of what might be called *decision support engineering*. I share Manola's view that *engineering* is a perfectly good term; however, those who do not like it can use *analysis* as an alternative.

Participants from countries with diverse forms of economic organization have demonstrated clearly that decision support

problems are independent of a particular overall economic structure—whether market- or planning-oriented. Emphases may vary in different economies, but even this is not clear. We have seen, for example, that the need for decision support systems to support individual and group deliberations at various levels in an organization and to support intraorganization communication is common to both East and West. Furthermore, there is everywhere currently inexorable technological progress in hardware development—and hence in software and communications development. This progress is partially the result of a common realization that we must rigorously control more complex systems in the face of dwindling resources and other factors. There is no question of stopping technological improvement; it will advance whether or not the participants in this conference have any part in it.

RESEARCH ISSUES

Among more specific points brought out at the meeting was the recognition that the misguided technological thrust in management and information sciences during the last two or three decades has been caused in part by our poor understanding of organizational behavior, of the management task environment, and of adult human learning. Understanding of all three is crucial to developing a better model for decision support engineering than the largely static paradigm evident thus far.

Most of operations analysis and management science (which developed from microeconomics, which in turn developed from physics) takes as given a static—or at least a temporally autonomous—paradigm. Certainly there are dynamic linear programming models, dynamic programming models, and optimal growth and planning models, but these are marginal extensions of classical physical dynamics applied to organizational problems. No one appears to have looked in depth at the mechanics of the dynamics of organizations, in the sense of human beings interacting with each other.

At this meeting we have seen a useful beginning in the related theories of Flores, Boxer, and R. Lee. The fundamental idea that we must understand the temporal sequence of actors' promises and requests—probably supported by analyses and decisions—is a useful way to look at the dynamic paths of human organizations. This notion gives an intrinsic definition of the organization as that which delimits allowable promises and requests (within the law). Further, it affords an immediate coupling of theories of incentives to individual, group, and organizational performance. From the point of view of decision support engineering, the roles of DSS generalist facilitators, organizational "gatekeepers" who watch over the introduction of new technology, risk takers, and risk averters, as well as the inefficacy of third-party project funding, will be more easily understood. This new view of organizational behavior is ripe for theoretical development, which can lead rapidly to the practical understanding crucial to decision support engineering. R. Lee's formalization of financial contracts is a first step in

this development. I think that the theory will not be independent of notions of hierarchy and temporal evolution. Indeed, what is striking about this general theory of human interaction is precisely that it does put a fundamental temporal dimension into the study of human organizations.

Psychologists generally agree that a theory of adult learning is thus far incomplete. We might hope that the carefully analyzed experience of decision support engineering will ultimately contribute to a largely nonexistent theory of adult learning, for coming to grips with applications usually generates theoretical insights. Thus decision support engineering could play a role in the advancement of more general human knowledge.

We appear to agree that whatever decision support systems are—and whatever decision support engineering is about—they are only part of the full range of information systems. This raises the question implicitly addressed at the meeting: Which part of the support of human beings with information technology is the concern of decision support engineering? It is obviously the part that is adaptive, that is underspecified, and that recognizes continually that complexity and perplexity are the order of the day. Decision support systems must be resilient in the face of changes—not only in the organizational environment, but also in the organizational structure, decision-making norms, and operating personnel. The necessity for user orientation, reconfiguration, and useful documentation is obvious.

We have seen at this meeting at least an indication of structures to solve the resiliency problem. Although different terminology has been used, there is a common underlying idea, which is perhaps best termed (after Sprague) a *DSS generator*. The basic idea is modularity of software, and possibly even of hardware—the DSS machine—perhaps especially in relation to committee support, the electronic board room, communications, and so on. Through the concept of the DSS generator, Keen's distinction between DSS *user* and DSS *builder* must ultimately merge. In this regard we should bear in mind Earnest's caveat that simple systems are sufficient for most tasks for which they will be used. Average versus worst-case performance analysis, design from simple particular to all embracing general, and the interaction of continual evolution with model definition stressed by Huber have been overlooked all too often. Iterative requirements analysis, quick and dirty "breadboard" systems, and Courbon's *approche evolutif* will always play a crucial role in decision support engineering. McLean's case history, however, shows that a facile approach to these ideas, even with top management support, is doomed to failure.

Decision support engineering is not (it seems easier to define what it is not) either purely model-oriented, i.e., focused on fixed static structures that can be used repetitively, or purely data-oriented. This, of course, does not mean that models and data are not integral parts of mature decision support systems, but DSS should be much more than just models and data. Indeed, decision support engineering must be more than the sum

of its reference disciplines. However, as Sagalowicz has pointed out from the perspective of one of them ("experimental epistemology," i.e., artificial intelligence), there are formidable difficulties in this fusion: problems, such as semantic models of database contents, remain to be resolved in each. We may hope that as well as posing challenging new problems to its reference disciplines, decision support engineering can contribute to the solution of old ones.

PROBLEM AREAS

Several problem areas that to my knowledge have not been addressed previously in computer-based operations research or database management systems arise in the context of decision support systems.

The first concerns probabilistic and time series (particularly judgmental) data and its handling. Methods of simple record data handling developed for transactions processing are not adequate to the data manipulation requirements of complex model solution procedures, such as time series analyzers and optimization procedures, operating on the reference database of a decision support system. In particular, we must develop methods for flagging sources, reliability, and uses of judgmental data in a decision support system to prevent the inevitable "hardening" of judgmental data as it moves up and out of an organization. (A. Lee has suggested that proven military intelligence practice may bear investigation in this context.)

Another important idea for development is that a decision support system should itself learn as the user learns. The software should have the facility to remove structured tasks as they are discovered from the user's responsibility to automatic treatment within the decision support system. There exist, for example, special-purpose database management systems (DBMS) that have this feature through a facility for "passive" commands; these may be built up to arbitrary complexity by nesting and are automatically executed on entry to appropriate system modules. Closely related is the property possessed by some recent DBMS (cf. Manola) of automatic database reorganization in light of query frequency. Some artificial intelligence systems have now been supplied with subsystems that query expert users with a view to making their knowledge automatically available to ordinary users. (This possibility of course depends on the ability to represent knowledge in the system as a set of formal rules.) Thus the rudiments of nontrivial system learning facilities are already being engineered; we should try to promote the development of these general ideas.

The next problem area, already mentioned, should be emphasized because it fits decision support engineering into the full range of information systems that support human organization. I use the term *organization* rather than *decision* because, as noted previously, decisions themselves are only part of the province of decision support engineering. Further, the scope of this field has been emphasized from a wide base of national interest at this

meeting. Some of the organizers of the meeting—including myself—believed that decision support engineering was concerned primarily with personal computing. This view is clearly nonsense. We must in fact think carefully not only about personal decision support systems, but also about group and committee decision support systems, organizational decision support systems, how these systems communicate, how tasks enter the support networks, and so on. Earnest's description of computerized support of the Artificial Intelligence Laboratory at Stanford University is an interesting beginning. His report on loss of temper—flaming—by participants in text-based electronic communication (undoubtedly partly caused by reduction in the range of the signal dimension present in visual or face-to-face communication) merits further investigation in the context of the electronic office. Current text-based communication also has lower response time than that of electronic implementation. Automatic response-forcing facilities in such electronic communication are a related area for research.

Finally—and this problem area is extremely important from the point of view of "marketing" decision support engineering—we must come to grips with the evaluation issue for decision support systems. Marketing is equally important in nonmarket economies, as people from those economies have pointed out. In all economies there are interest groups and there are those trying to "sell" decision support systems. Some constructive suggestions regarding system evaluation have been made at this meeting. Perhaps the most important concern various kinds of use measures, which prompted much discussion, both implicit and explicit. As we have been emphasizing that decision support systems are adaptive (they must change), the number of change requests, in particular, must be a crucial use measure. Others include the numbers of system queries and new users. Such measures imply the necessity for careful classification and frequency analysis of tasks and users in a given decision support implementation.

CONCLUSION

While collectively we have not given detailed direction for the field for the next five years, two years, or even one year, we have made significant progress in indicating some future tasks. I hope that conference participants will take with them their individual learning experiences and will contribute to the practice, research, teaching, and promotion of decision support engineering. We have put forth a challenge to develop an effective methodology for identifying and documenting successful—and unsuccessful—decision support system implementations. Equally difficult is the question of how to teach decision support engineering. More straightforward is the need to promote the subject and practice in information technology trade journals and in business magazines. Addressing these general issues will help to promote an important subject.

APPENDIX 1: LIST OF PARTICIPANTS

A.J. Barbarie
Management and Technology Area
IIASA
Schloss Laxenburg
A-2361 Laxenburg
Austria

P.J. Boxer
London Graduate School of Business Studies
Center for Management Development
Sussex Place
London NW1 4SA
United Kingdom

V.B. Britkov
All-Union Research Institute for Systems Studies (VNIISI)
29, Ryleyev Street
Moscow, 119034
USSR

M.A.H. Dempster
System and Decision Sciences Area
IIASA
Schloss Laxenburg
A-2361 Laxenburg
Austria

G. Dickson
College of Business Administration
University of Minnesota
271 19th Ave. S.
Minneapolis, Minnesota 55455
USA

L. Earnest
Stanford Artificial Intelligence Laboratory
Stanford University
Palo Alto, California 94306
USA

G. Fick
Management and Technology Area
IIASA
Schloss Laxenburg
A-2361 Laxenburg
Austria

C.F. Flores
Computer Science Department
Stanford University
Stanford, California 94305
USA

R. Hackathorn
Division of Information
Sciences Research
College of Business
Administration
University of Colorado
Campus Box 419
Boulder, Colorado 80309
USA

P. Harris
Systems Analysis Research Unit
Department of the Environment
2 Marsham Street
London SW1P 3EB
United Kingdom

J.H. Hawgood
Computer Unit
Science Laboratories
University of Durham
South Road
Durham DH1 3LE
United Kingdom

P. Hearson
Central Computer and Tele-
communications Agency
Civil Service Department
Riverwalk House
157-161 Millbank
London SW1P 4RT
United Kingdom

G.P. Huber
Graduate School of Business
University of Wisconsin
1155 Observatory Drive
Madison, Wisconsin 53706
USA

H. Huebner
Institut fuer Unternehmens-
fuehrung
Universitaet Innsbruck
Blasius-Hueber-Strasse 16/2
A-6020 Innsbruck
Austria

P.G.W. Keen
The Sloan School of Management
The Massachusetts Institute
of Technology
50 Memorial Drive
Cambridge, Massachusetts 02129
USA

G.Kochetkov
The USA and Canada Institute
of The USSR Academy of Sciences
Khelbny Per. 2/3
Moscow, G-69
USSR

Alec Lee
Management and Technology Area
IIASA
Schloss Laxenburg
A-2361 Laxenburg
Austria

Ronald M. Lee
Management and Technology Area
IIASA
Schloss Laxenburg
A-2361 Laxenburg
Austria

F. Manola
Computer Corporation of America
575 Technology Square
Cambridge, Massachusetts 02139
USA

L.B. Methlie
Institute for Information
Systems Research
Norwegian School of Economics
and Business Administration
Helleveien 30
N-5000 Bergen
Norway

E.R. McLean
Graduate School of Management
University of California at
Los Angeles
Los Angeles, California 90024
USA

G. Mueller
IBM Scientific Center
Tievgartenstr. 15
D-6900 Heidelberg
FRG

E.A. Nurminski
System and Decision Sciences Area
IIASA
Schloss Laxenburg
A-2361 Laxenburg
Austria

M.M.L. Pearson
General Research Area
IIASA
Schloss Laxenburg
A-2361 Laxenburg
Austria

S. Persson
Handelshogskolan
Box 6501
S-11383 Stockholm
Sweden

D. Sagalowicz
SRI International
333 Ravenswood Ave.
Menlo Park, California 94025
USA

R.L. Sheinin
All-Union Research Institute for Systems Studies (VNIISI)
29, Ryleyev Street
Moscow, 119034
USSR

H.G. Sol
Information Systems Research Group
Faculty of Economics
University of Groningen
P.O. Box 800
9700 AV Groningen
The Netherlands

R.H. Sprague, Jr.
Department of Decision Sciences
College of Business Administration
University of Hawaii at Manoa
2404 Maile Way
Honolulu, Hawaii 96822
USA

C.B. Stabell
Institute for Information Systems Research
Norwegian School of Economics and Business Administration
Helleveien 30
N-5000 Bergen
Norway

A. Ter-Manuelianc
Institute of Management
115 49 Prague 1
Jungmannova 29
CSSR

B.L. Trippett
NCR Corporation
NCR Flat
121-122 Berkeley Court
Glentworth Street
London NW1
United Kingdom

A. Vari
Bureau for Systems Analysis at the State Office for Technical Development
H-1374 Budapest 5, P.O.B. 565
Hungary

G.R. Wagner
Execucom Systems Corporation
P.O. Box 9758
Austin, Texas 78766
USA

APPENDIX 2: DISCUSSION GROUPS

Group 1: Gary Dickson, Chairman
Heinz Huebner, Rapporteur

Les Earnest
Göran Fick
Peter Harris
George Huber
Frank Manola
Roman Sheinin
Anna Vari

Group 2: Leif Methlie, Chairman
Alain Barbarie, Rapporteur

Vladimir Britkov
John Hawgood
Peter Keen
Ephraim McLean
Evgenji Nurminski
Daniel Sagalowicz

Group 3: G.R. Wagner, Chairman
Ronald Lee, Rapporteur

Michael Dempster
Guenter Mueller
Michael Pearson
Ralph Sprague
Henk Sol
Antonin Ter-Manuelianc

Group 4: Dick Hackathorn, Chairman
Phillip Boxer, Rapporteur

Fernando Flores
Peter Hearson
Gennady Kochetkov
Staffan Persson
Charles Stabell
Bernard Trippett

APPENDIX 3: TASK FORCE ORGANIZATION

Co-Chairmen	Göran Fick Ralph H. Sprague, Jr.
Program Committee:	Alain Barbarie Michael A.H. Dempster Göran Fick Richard Hackathorn Peter Keen Alec M. Lee Leif B. Methlie Ralph H. Sprague, Jr.
Executive Secretary:	Miyoko Yamada
Local Arrangements:	Gabriele Adam
Communications Support:	The University of Hawaii and the Annual Hawaii International Conference on System Sciences supported the computer-based communication that was used to plan and coordinate this meeting.
Proceedings:	The editors would like to acknowledge the accurate, extensive, and fast work done by Lory Hervey of the IIASA Editorial Group in preparing the manuscript and at all stages of the production of the proceedings.

APPENDIX 4: LIST OF ACRONYMS

AI	Artificial Intelligence
ACM	Association for Computing Machinery
AFIPS	American Federation of Information Processing Societies
APL	A Programming Language
CAI	Computer-aided Instruction
CMS	Computer-based Management System
CPM	Critical Path Method
DB/DC	Data Base/Data Communications
DBMS	Data Base Management System
DDBMS	Distributed Database Management System
DD/D	Data Dictionary/Directory
DDL	Data Description Language
DGMS	Dialogue Generation and Management Software
DML	Data Manipulation Language
DP	Data Processing
DSA	Decision Support Analysis
DSE	Decision Support Engineering (Engineer)
DSS	Decision Support System
EDP	Electronic Data Processing
EIS	Executive Information System
GSS	Group Support System
HW	Hardware
IFPS	Interactive Financial Planning System
IS	Information System
MAPP	Managerial Analysis for Profit Planning
MBMS	Model Base Management Software
MIS	Management Information System
MS	Management Science
NCC	National Computer Conference
OA	Office Automation
OR	Operations Research
OSS	Organizational Support System
OW	Orgware (organizational setting, procedures, etc.)
PBX	Private Branch Telephone Exchange
PERT	Program Evaluation and Review Technique
SW	Software
WP	Word Processing